ADDRESS BOOK CLEARINGHOUSE INTERFACE SYSTEM AND METHOD

FIELD OF THE INVENTION

The present invention relates in general to computerized address books and in particular to a system and method for providing access to an integrated address book clearinghouse.

BACKGROUND OF THE INVENTION

Computerized address books and personal information managers are well known in the computing field. Software applications such as email programs, instant messaging programs and computer fax programs include or have access to computerized address books of one form or another. Such conventional address book utilizing applications generally store address book information locally on a computing device. Often the address books are in incompatible formats, requiring that the information contained in each application's address book be individually keyed in. Even where some of the format fields are common, common information must often be entered multiple times, once for each address book. Some applications have integrated address book in a way that allows a personal information manager, that can send email, stores physical address information along with email addresses. Other address book integration examples include email program address books that include instant messenger handles or fax program address books that include physical addresses and voice phone numbers.

While integrated address books included in or accessible by applications running on client computers have solved some of the previous problems of having to enter identical or

similar information multiple times (once for each different type of program), such address books are ineffective when applications are located on remote computers connected to client computers or devices via a network. More specifically, networks are also well known in the computing field. By definition, a network is a group of computers and associated devices that are connected by communication facilities or links. An internetwork, in turn, is the joining of multiple computer networks, both similar and dissimilar, by means of gateways or routers that facilitate data transfer and conversion from the multiple computer networks. A well known abbreviation for the term internetwork is "internet." As currently understood, the capitalized term "Internet" refers to the collection of networks and routers that use the Internet protocol to communicate with one another. The Internet has recently seen increased growth by virtue of its ability to link computers based throughout the world. As will be better appreciated from the following description, the present invention could find use in many network environments. However, for purposes of discussion, the Internet is used as an exemplary network environment for implementing the present invention.

The Internet has quickly become a popular method of disseminating information due, in large part, to its ability to deliver information quickly and reliably. To retrieve online resources or other data over the network, a user typically uses communications or network browsing software. A common way of retrieving online resources is to use such communications or network browsing software to access online resources at a uniform resource identifier ("URI") address such as a uniform resource locator ("URL") address that indicates the location of the online resources on a server connected to the network or internetwork.

As the Internet (and other networks) have developed some of the address book utilizing applications that were formerly performed on client devices are now provided by online resources accessed via the network. One example is a Web-based email network application. As a result, the storage of address book information was shifted from client devices to online accessible devices. Online address book information storage eliminates the need for a user to export address book information when the user changes to a new device and/or adds a new device to the user's inventory of devices.

Prior attempts have been made to integrate separate address book information associated with network applications. One example is an integrated online contact list that compares all addresses in an online contact list to the addresses in an online buddy list to determine which contacts in the contact list should have an indicator showing that the contact has its instant messaging capabilities enabled (contacts with instant messaging capabilities are commonly referred to as "buddies"). This system is inefficient and stores unnecessarily redundant information. More specifically, this system retains two separate lists, an online contact list and an online buddy list, each of which can be separately updated. If inconsistent information is added to either list, their comparisons and pairings break down. For example, if the email address of a contact in the address book of a Web based email application was different than the email address of a buddy in the address book of a Web based instant messenger application, the pairing between the two is broken unless some other type of link is used to maintain the pairing.

Even though such prior attempts allowed for secure access to a user's network-based application's address book information, the information was not available to an authenticated user in an integrated fashion.

Accordingly, there is a need for an address book information clearinghouse, and an interface thereto, that provides secure access and management of address book information for particular users to multiple client device applications and/or network-based applications. It is desirable that any address book clearinghouse interface accept and provide information in an application independent manner.

SUMMARY OF THE INVENTION

The present invention is directed to a programming interface, system, and computer-readable medium for accessing and managing an integrated online address book clearinghouse. One aspect of the present invention provides a programming interface layer with address book management functions for accessing and managing an integrated online address book clearinghouse. The programming interface layer receives application programming interface ("API" or programming interface) calls from applications desiring access and management of address books included in the integrated address book clearinghouse. The programming interface layer executes one or more functions in response

to received programming interface calls. The address book functions manage address books by adding, deleting, updating, and finding address books, contacts, and groups of contacts in the integrated address book clearinghouse. Additionally, the functions can manage communications permissions for address books that have communications limitations (e.g., address books for applications with parental control limits or employer-imposed limits). The programming interface layer also uses identity information and any associated identity authentication information to assure that each API call is directed to a desired address book when an address book function is executed. The programming interface layer further includes a parameter processing module for processing function-specific parameters passed in data envelopes to the programming interface layer by the applications. The programming interface layer also includes, in this aspect, a response generating module to generate a function-specific response from an address book function that is encapsulated in a data envelope and sent back to the application that sent an API call to the programming interface layer.

In some aspects of the present invention, the authentication information associated with identity information is explicitly included as a parameter passed in a data envelope to the programming interface layer.

As can be seen from the summary above, the present invention provides a programming interface for accessing and managing an online integrated address book clearinghouse system and a related computer-readable medium and system.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIGURE 1 is a pictorial diagram of an exemplary, simplified, integrated online address book clearinghouse;

FIGURE 2 is a block diagram of a public front end server suitable for use in FIGURE 1;

FIGURE 3 is a diagram illustrating the actions of a client device, authentication server, public front end server, and an address book clearinghouse storage when accessing the online address book clearinghouse shown in FIGURE 1;

FIGURE 4 is a flow diagram illustrating an address book clearinghouse API calling routine according to the present invention;

FIGURE 5 is a flow diagram illustrating an address book clearinghouse API call processing routine according to the present invention;

FIGURE 6 is a flow diagram illustrating an address book creation subroutine suitable for use in the address book clearinghouse call processing routine illustrated in FIGURE 5;

FIGURE 7 is a flow diagram illustrating an alternate address book clearinghouse API calling routine according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The detailed description which follows is represented largely in terms of processes and symbolic representations of operations by conventional computing components including processors, memory storage devices for the processors, connected display devices, and input devices, all of which are well known in the art. These processes and operations may utilize conventional computing components in a heterogeneous distributed computing environment, including remote storage servers, computer servers, and memory storage devices; such processes, devices, and operations also being known to those skilled in the art and others. Each of these conventional distributed computing components is accessible by the processors via a communications network.

The present invention is directed to providing a programming interface for managing network accessible address books embodied in an integrated online address book clearinghouse. The programming interface layer in one exemplary embodiment of the present invention is an application programming interface ("API") with specific address book clearinghouse functions for managing a multitude of network accessible address books shared across multiple devices and applications. One exemplary embodiment of such an API is described in the attached appendix. However, those of ordinary skill in the art and others will appreciate that the attached API description is merely one example of a programming

interface suitable for accessing an integrated online address book clearinghouse system and that, within the scope and spirit of the present invention, other APIs are possible.

FIGURE 1 is a pictorial diagram of an exemplary, simplified integrated online address book clearinghouse 100 for providing address book information (address books, contacts, contact groups, etc.) associated with particular users to client devices via the Internet 105. For ease of illustration, a single client device 110 shown pictorially as a personal computer is included in FIGURE 1, it being recognized that in actual implementations of the multiple client devices in a variety of forms, including cellular telephones and personal digital assistants (PDAs), would be included. That is, the integrated address book clearinghouse 100 functions in a distributed computing environment that includes a plurality of computing devices interconnected by the Internet 105 (or some other suitable network). In addition to the client device 110, the plurality of computing devices shown in FIGURE 1 include an authentication server 115 and a partner server 120. The address books clearinghouse includes a partner front end server 130, a public front end server 200, and an address book clearinghouse storage 150. The client device 110 has computing capabilities and may be any form of device capable of communicating with the other devices of the present invention. As noted above, while the client device 110 is pictorially shown as a personal computer, this depiction should be taken as illustrative and not limiting. As will be appreciated by those of ordinary skill in the art, the authentication server 115, partner server 120, partner front end server 130, public front end server 200 and address book clearinghouse storage 150 may reside on any device accessible, directly or indirectly, by the client device 110, shown in FIGURE 1 via the Internet 105. An exemplary public front end server 200 is shown in further detail in FIGURE 2 and described below.

While only a single client device 110, authentication server 115, partner server 120, partner front end server 130, public front end server 200, and address book clearinghouse storage 150 have been shown in FIGURE 1, it will be appreciated that many more client devices 110, authentication servers 115, partner servers 120, partner front end servers 130, public front end servers 200, and address book clearinghouse storages 150 can be included in an actual system practicing the present invention. The devices of the address book clearinghouse, namely, the public front end server 200, partner front end server 130, and

address book clearinghouse 150 provide access to integrated address book information stored in the address book clearinghouse storage 150. Additionally, while both a public front end server 200 and partner front end server 130 are shown in FIGURE 1, it will be appreciated that in alternate embodiments of the present invention, only a public front end server 200 or only a partner front end server 130 may be used without departing from the spirit and scope of the present invention.

FIGURE 2 illustrates an exemplary public front end server 200 suitable for use in the address book clearing house 100 shown in FIGURE 1. In its most basic form, the public front end server 200 typically includes at least one processing unit 202 and memory 204. Depending on the exact configuration and type of public front end server 200, memory 204 may be volatile (such as RAM), nonvolatile (such as ROM, flash memory, etc.), or some combination of the two. This most basic configuration is illustrated in FIGURE 2 by dashed line 206. The public front end server 200 may also have additional features and/or functionality. For example, the public front end server 200 may also include additional storage (removable and/or nonremovable) including, but not limited to, magnetic or optical discs or tape. Such additional storage is illustrated in FIGURE 2 by removable storage 208 and nonremovable storage 210. In general, computer storage media includes volatile and nonvolatile, removable and nonremovable media implemented in any method or technology for storage of computing information (e.g., computer-readable instructions, data structures, program modules, other data, etc.). Memory 204, removable storage 208, and nonremovable storage 210, are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory, or other memory technology, CD, DVD, or other optical storage, magnetic cassettes, magnetic tape, magnetic disc storage, or other magnetic storage devices, or any other medium which can be used to store or read desired information and which can be accessed by the public front end server 200. Any such computer storage media may be part of the public front end server 200. The memory 204 of a public front end server 200 practicing the present invention stores an programming interface (API) layer 220 that includes routines for accessing storage resources.

The public front end server 200 also contains a communications connection 212 that allows the device to communicate with other devices. The communications connection 212

is used to communicate computer-readable instructions, data structures, program modules or other data using a modulated data signal that includes a carrier wave or other transport mechanism modulated by the data to be communicated. By way of example and not limitation, communication connection 212 includes wired connections, both copper and optical, and wireless connections such as acoustic, radio frequency, infrared, etc. Public front end server 200 may also have input device(s) 214 such as a keyboard, a mouse, a pen, a voice input device, a touch input device, etc. Output device(s) 216 such as a display, speakers, a printer, etc., may also be included. Since all these devices are well known in the art, they are not described here.

Since, in general, the partner front end server 130 can be similar to the public front end server 200 described above, the partner front end server 130 is not described in detail. Similarly, the authentication server 115 and the partner server 120, except for the API Layer 222, can also be similar to the public front end server 200 described above, and these servers are also not described in detail.

The operation of an integrated address book clearinghouse 100 according to the present invention will be better understood by reference to FIGURE 3, which illustrates one exemplary sequence of communication interactions between the client device and selected ones of the servers shown in FIGURE 1, and the address book clearinghouse storage 150. The selected servers are the authentication server 115, and the public front end server 200. The exemplary communications interactions shown in FIGURE 3 begin by the client device 110 sending 301 an encapsulated address book API call and data envelope to the public front end server 200. Next, the public front end server 200 parses 303 the data envelope to retrieve the API call and any function parameters. In the exemplary communications interactions shown in FIGURE 3, the public front end server 200 redirects 305 the client device 110 to the authentication server 115 if the authentication data in the envelope is expired or missing. If this occurs, the client device 110 submits 310 an authentication request to the authentication server 115. Those of ordinary skill in the art and others will appreciate that authentication servers 115 may take many forms. In one exemplary embodiment of the present invention the authentication server 115 is provided by a network authentication service such as Microsoft's .NET Passport service. Other

authentication servers 115 able to verify a user's identity using conventional authentication techniques may be used as well.

The authentication server 115 authenticates 315 the identity included in the authentication request 310. After the authentication server has authenticated 315 the identity, authentication credentials are sent 320 back to the client device 110. The authentication credentials may take any form that can be provided by the client device 110 to other devices to indicate that client device 110 is being operated by a user who is allowed to access the address book clearinghouse server 150 via the public front end server 200. One exemplary form of authentication credentials is authentication "cookies." An authentication cookie is cryptographically signed and encrypted data provided to a client device 110 by an authentication server115 to persist an identity's authentication at the client device 100. Other exemplary forms of authentication credentials include kerberos tokens and transient cryptographically signed and authenticated access information.

Next, the client device 110 encapsulates 325 parameters in an address book function data envelope that includes the authentication credentials and other address book function specific parameters. This further (second) encapsulated address book API call and data envelope is then sent 330 to the public front end server 200. The public front end server 200 parses 335 the envelope. Next, the public front end server 200 verifies 340 the encapsulated authentication credentials included in the data envelope. Once the identity of the user is thusly verified, the public front end server sends 345 an clearinghouse function call to the address book clearinghouse storage 150 that corresponds to the address book function specified in the API call. The address book clearinghouse storage 150 processes 350 the clearinghouse function call and returns 355 the function call response to the public front end server 200. The public front end server 200 then encapsulates 360 the function call response in a response data envelope that identifies the API call it is responding to and the identified address book of that call and returns 365 the encapsulated API response data envelope to the client device 110.

Those of ordinary skill in the art and others will appreciate that the communications interactions illustrated in FIGURE 3 provide structured and efficient access to an online address book clearinghouse. If necessary, the request for access can be authenticated prior to

allowing access to the address book clearinghouse storage 150. Those of ordinary skill in the art and others will appreciate that FIGURE 3 represents only one exemplary set of communication interactions. Others are possible. For example, authenticating an identity by the authentication server 115 may grant the authenticated user permission to send other identities to the public front end server 200, e.g., may grant a parent permission to access address books belonging to a child associated with the parent (and their identity). Address books with parental controlled access and other forms of associated access (such as employer controlled access) are allow one user to control the accesses and communication of another user. Exemplary "ABAllowListSet" (for setting those emails and instant messenger contacts that a particular subsidiary user has access to) and "ABAllowListGet" (for setting the emails and instant messenger contacts that a particular subsidiary user is allowed to receive from) functions are described in greater detail in the Appendix portion of this detailed description.

The communication interactions illustrated in FIGURE 3 between various external devices and devices of the integrated address book clearinghouse 100 may employ any conventional computing communication form. In one exemplary embodiment of the present invention, the communications are formatted using the Simple Open Access Protocol ("SOAP") with extensible markup language ("XML") formatted instructions and/or parameters. An exemplary XML formatted instruction for an address book creation request (ABAdd) is illustrated by the following code:

TABLE

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.msn.com/webservices/AddressBook/ABAdd"

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <ABApplicationHeader xmlns="http://www.msn.com/webservices/AddressBook">
      <ApplicationId>00000000000000000000010010efd4e487</ApplicationId>
      <IsMigration>0</IsMigration>
    </ABApplicationHeader>
    <ABAuthHeader xmlns="http://www.msn.com/webservices/AddressBook">
      <ManagedGroupRequest>0</ManagedGroupRequest>
    </ABAuthHeader>
  </soap:Header>
  <soap:Body>
    <ABAdd xmlns="http://www.msn.com/webservices/AddressBook">
      <abInfo>
        <name></name>
        <ownerEmail>ken@hotmail.com</ownerEmail>
        <fDefault>1</fDefault>
        <isMessengerMigrated>0</isMessengerMigrated>
      </abInfo>
    </ABAdd>
  </soap:Body>
</soap:Envelope>
```

Those of ordinary skill in the art and others will appreciate that the address book creation request (ABAdd) illustrated above is merely one form of instructions between the FIGURE 3 devices and that many other forms of communications may be possible. The attached appendix includes other forms of data envelopes and methods that may be used to communicate between the FIGURE 3 devices.

FIGURE 4 is a flow chart illustrating an address book clearinghouse API call routine 400 implemented by the client device 110 for carrying out the communication interactions shown in FIGURE 3. The API calling routine 400 begins at block 401 and proceeds to block 405 where parameters for an address book function are encapsulated in an address book API call and data envelope by the calling client device. The data envelope is then sent in block 410 to the public front end server 200. Next, in block 415 an address book function-specific response is received back from the public front end server 200 to the encapsulated address book API call and data envelope sent in block 405. The address book clearinghouse API calling routine 400 then ends at block 499.

After the public front end server 200 receives the address book API call and data envelope sent by the client device as shown in FIGURE 3, the public front end server 200 processes the API call. FIGURE 5 illustrates an API call processing routine 500. The API call processing routine 500 begins at block 501 and proceeds to block 505 where both the address book API call and data envelope is received from the client device 110. Next, the data envelope is parsed in block 510 to extract the API call parameters, including the authentication data associated with an address book function being accessed by the API call. The parsing of the data envelope (e.g. by executing an parameter processing module) also identifies which address book function is being called by the API. In block 515, the authentication information in the data envelope is verified (e.g. by executing an identity authentication module). In decision block 520 a determination is made whether the identity identified by the authentication information is authorized to access the called address book function. If the identity is not authorized, processing proceeds to block 560 where an error condition is generated as an address book function response. Processing then continues to block 545 where the address book function response is encapsulated in a data envelope (e.g. by executing an response generating module). In block 550, the data envelope is sent to the client device, after which API call processing routine 500 ends at block 599.

As will be readily understood by those skilled in the art and others, the path through blocks 560, 545 and 550 may cause the client device to seek authorization credentials for an authorization server 115 as shown in FIGURE 3 and described above.

If in decision block 520 a determination was made that the identity is authorized, then processing proceeds to decision block 530 where a determination is made whether an address book exists for the identity. If no address book is found to exist in decision block 530, processing proceeds to subroutine block 600 where an address book is created for the identity. An address book creating subroutine 600 is illustrated in FIGURE 6 and described below. After an address book has been created, and the address book creating subroutine 600 has returned, processing proceeds to block 535. If an address book was found to exist in decision block 530, processing proceeds directly to block 535. At block 535 the clearinghouse function call specified in the received data envelope is sent to the address book clearinghouse storage 150. In block 540, the address book function response is received from the address book clearinghouse storage. Processing then proceeds to block 545 where the address book function response is encapsulated in a data envelope as discussed above. Then, in block 550 the API response data envelope is sent to the client device.

Those of ordinary skill in the art and others will appreciate that there are many possible API function calls that may be made to an address book clearinghouse. The appendix to this detailed description, includes a number of exemplary address book clearinghouse API function calls. The exemplary API function calls include: ABAdd (for adding a new address book); ABDelete (for deleting an existing address book); ABUpdate (for updating an address book); ABFind (for locating an address book); ABContactAdd (for adding a new contact); ABContactDelete (for deleting an existing contact); ABContactUpdate (for updating a contact); ABContactFind (for finding Contacts), ABFindAll (for gathering all address book, contact and group information); ABFindByContacts (for finding all contacts and any groups containing listed contacts); ABFindByGroups (for locating all contacts in given groups and the groups themselves); ABFindMessengerUsers (for locating contacts with instant messaging capability); ABFindMeContact (for locating the contact for the address book owner); ABGroupAdd (for adding a new group); ABGroupDelete (for deleting an existing group); ABGroupUpdate (for updating a group); ABGroupFind (for locating a group); ABGroupContactAdd (for adding a contact to a group); ABGroupContactDelete (for removing a contact from a group);

ABAllowListSet (for setting those emails and instant messenger contacts that a particular subsidiary user has access to); ABAllowListGet (for setting the emails and instant messenger contacts that a particular subsidiary user is allowed to receive from). Those of ordinary skill in the art and others will appreciate that both more and less API function calls may be employed in an address book clearinghouse system, without departing from the spirit and scope of the present invention.

FIGURE 6 illustrates an exemplary address book creating subroutine 600 for creating an identity coupled address book in accordance with the present invention. The address book creating subroutine 600 begins at block 601 and proceeds to block 605 where an address book creation request is received for a specific user identity. In block 610, the address book creation request is sent to the address book clearinghouse storage 150 indicating that an address book should be created for the specified identity. In block 615, an address book creation confirmation is received back from the address book clearinghouse storage. Address book creating subroutine 600 then ends at block 699, returning the address book identification to its calling routine.

Those of ordinary skill in the art and others will appreciate that address book creating subroutines may include other elements and steps other than those listed above and illustrated in FIGURE 6. The "ABAdd" API call listed in the appendix provides exemplary parameters that may be used when creating an address book in accordance with the present invention. The ABAdd API call is presented merely as one example of an address book creating function call, and those of ordinary skill in the art and others will appreciate that there are a myriad of other possible address book creating function calls all of which fall within the spirit and scope of the present invention.

FIGURE 1 illustrates a partner server 120 in communication with devices of the integrated address book clearinghouse 100. A flow chart illustrating an address book clearinghouse API calling routine 800 implemented by the partner server 120, in accordance with one exemplary embodiment of the present invention, is shown in FIGURE 7. The API calling routine 700 begins at block 701 and proceeds to block 705 where a function call request is received from the client device 110. Next, in block 710, parameters for an address book function are encapsulated in an address book function-specific data envelope. The data

envelope is then sent in block 715 to the partner front end server 120. Next, in block 720 an address book function-specific response is received back from the partner front end server 120 that corresponds to the encapsulated address book function-specific data envelope sent in block 710. In block 725, the partner server 120 sends a partner response to the client device 110 with information from the address book function-specific response received from the partner front end server 120. Address book clearinghouse API calling routine 700 then ends at block 799.

Those of ordinary skill in the art and others will appreciate that the API calling routine of the partner server 120 illustrated in FIGURE 7 is similar to the API calling routine of the client device 110 illustrated in FIGURE 4. It will also be appreciated that various actions delegated to the client device 110 or the partner server 120 may be performed in whole or in part by the client device 110 or the partner server 120, when a partner server (or other intermediary devices) is interposed between the partner front end server 130 and the client device 110 when accessing the address book clearinghouse storage 150.

While the presently preferred embodiment of the invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

APPENDIX

A. Objects

A(i). Address Book

Address Book Properties

References to Guid or GUID refer to a globally unique identifier, such as "System.Guid" (See the .NET Framework SDK for more information).

References to Puid or PUID refer to a globally unique identifier used in authentication system, such as the Passport system.

Table:

```
<AB>
   <abId>guid</abId>
   <abInfo>
        <name>string</name>
        <ownerPuid>long</ownerPuid>
        <ownerEmail>string</ownerEmail>
        <fDefault>boolean</fDefault>
   </abInfo>
   <propertiesChanged>ABName</propertiesChanged>
   <createDate>dateTime</createDate>
   <lastChange>dateTime</lastChange>
</AB>
```

AB.abId: The Address Book service associates a unique GUID with each Address Book. This GUID is a zero filled PUID.

For the Passport Authed FE, the Identity may be zero. This is an indication that the PUID for the AB should be taken from the authentication header in the SOAP request.

(PUID Decimal: 281547719894151; PUID Hex: 0x10010efd4e487; and PUID zero filled abId: 00000000-0000-0000-0001-0010efd4e487)

AB.propertiesChanged: Only used in ABUpdate. Set by the caller to indicate which fields should be updated.

AB.createDate: The date the Address Book was created.

AB.lastChange: Format: GMT. ISO 8601 format. Purpose: Used by partners that keep a local cache of Groups and Contacts and need to synchronize with the Address Book.

ABInfo.name: 60 char max.

ABInfo.ownerPuid: PUID of the owner of the Address Book. Used when provisioning the Address Book. Will be zero over the Passport authed front end except for parental control scenarios. See ABAdd.

ABInfo.ownerEmail: Must be a valid SMTP address. Email address of the owner of the Address Book.

ABInfo.fDefault: Reserved for a future version

ABInfo.isMessengerMigrated: This flag is read-only to all partners except Messenger. You must have the Messenger application id to change this value. True - Messenger has migrated the Messenger Users to this account. False - No migration has occurred. On setup, this field should be seeded with false or null. After migration, the default for this field will be true for new Address Books.

A(ii). Group

A Group is a named collection of Contacts in the Address Book.

General rules:

- A Group MUST have a unique name within the Group type.
- A Group MAY have no Contacts.
- A single Contact MAY exist in multiple Groups.
- No nesting. A Group cannot contain another Group.
- No sharing. A Group cannot be shared with other Address Books currently.

Table: Group Properties

```
<Group>
   <groupId>guid</groupId>
   <groupInfo>
        <groupType>guid</groupType>
        <name>string</name>
        <clientErrorData>string</clientErrorData>
   </groupInfo>
   <propertiesChanged>GroupName</propertiesChanged>
   <fDeleted>boolean</fDeleted>
   <lastChange>dateTime</lastChange>
</Group>
```

Group.groupId: ABCH generated GUID used to identify this group.

Group.propertiesChanged: Only used in ABGroupUpdate. Set by the partner to indicate which fields should be updated.

Group.fDeleted: True if this Group was deleted from the Address Book.

Group.lastChange: Format: GMT. ISO 8601 format. Purpose: Used by partners that keep a local cache of Groups and Contacts and need to synchronize with the Address Book.

GroupInfo.groupType: MSNAB - the Address Book Group (default), MSNBUDDY - Messenger Users List Group, MSNPUBLIC - generic Groups ("Friends", "Soccer", ...), Value is passed as a GUID in the SOAP request.

GroupInfo.name: REQUIRED. Must be UNIQUE within the group type. Group names are not case-sensitive. You cannot add two Groups with names "Group1" and "GROUP1".

GroupInfo.quickOrder: Used to store the quicklist order for any group. Not maintained by the Address Book. Field is nullable. It is 2 bytes so that it can be used to order up to 64k contacts.

GroupInfo.clientErrorData: Used by partners to identity the Group that failed during a bulk operation. Only needed when specifying multiple Groups in a request. This value is not stored in the ABCH. The value is only used during the request/response. The value is needed because a Group Id will not have been generated and the Group Name information (which would otherwise be unique) - may be incorrect or invalid.

Table:

```
<groupFilter>
  <groupIds>
       <guid>guid</guid>
       <guid>guid</guid>
  </groupIds>
  <groupTypes>
       <guid>guid</guid>
       <guid>guid</guid>
  </groupTypes>
</groupFilter>
```

groupIds: The Ids (GUIDs) of specific MSNPUBLIC Groups. For example, the Id of the "Carpool" Group. If the same GroupId is passed multiple times, it will only be returned once. If the groupId does not exist, no error will result

groupTypes: The MSNAB, MSNBUDDY, and MSNPUBLIC Groups are identified by type. Pass the GUID that represents each type. If the same GroupType is passed multiple times, it will only be returned once. If the groupType is invalid, no error will result.

A(iii). Contact

A Contact is an individual listed in the Address Book.

General rules: A Contact MAY exist in multiple Groups. A Contact MUST be identified with a Quickname. The Quickname and Passport fields MUST be unique. Note that Quicknames are not case-sensitive.

A Messenger User is defined as a Contact that: Has a value in the Passport field, Has the IsMessengerUser flag set, The number of Contacts that may be marked as isMessengerUser is constrained to 150.

Table: Contact Properties:

```
<Contact>
  <contactId>guid</contactId>
  <contactInfo>
       <contactType>Regular or Me</contactType>
       <quickName>string</quickName>
```

```
<firstName>string</firstName>
<lastName>string</lastName>
<passportName>string</passportName>
<puid>int64</puid>
<comment>string</comment>
<isMobileVisible>boolean</isMobileVisible>
<isMobileIMEnabled>boolean</isMobileIMEnabled>
<isMessengerUser>boolean</isMessengerUser>
<isFavorite>boolean</isFavorite>
<isSmtp>boolean</isSmtp>
<spotWatchState>NoMessaging</spotWatchState>
<birthdate>dateTime</birthdate>
<quickOrder>int16</quickOrder>
<primaryEmailType>
     ContactEmailPersonal or
     ContactEmailBusiness or
     ContactEmailOther
</primaryEmailType>
<emails>
     <ContactEmail>
          <contactEmailType>
               ContactEmailPersonal or
               ContactEmailBusiness or
               ContactEmailOther or
               ContactEmailMessenger
          </contactEmailType>
          <email>string</email>
     </ContactEmail>
</emails>
<phones>
     <ContactPhone>
          <number>string</number>
          <contactPhoneType>
               ContactPhonePersonal or
               ContactPhoneBusiness or
               ContactPhoneMobile or
               ContactPhonePager or
               ContactPhoneFax or
               ContactPhoneOther or
               ContactPhoneLivePersonal or
               ContactPhoneLiveBusiness or
               ContactPhoneLiveMobile
          </contactPhoneType>
     </ContactPhone>
</phones>
<locations>
     <ContactLocation>
          <contactLocationType>
               ContactLocationPersonal or
```

```
                    ContactLocationBusiness
                </contactLocationType>
                <name>string</name>
                <street>string</street>
                <city>string</city>
                <state>string</state>
                <country>string</country>
                <postalCode>string</postalCode>
            </ContactLocation>
        </locations>
        <webSites>
            <ContactWebSite>
                <webURL>string</webURL>
                <contactWebSiteType>
                    ContactWebSitePersonal or
                    ContactWebSiteBusiness
                </contactWebSiteType>
            </ContactWebSite>
        </webSites>
        <groupIds>
            <groupId>guid</groupId>
            <groupId>guid</groupId>
        </groupIds>
        <annotations>
            <Annotation>
                <Name>string</Name>
                <Value>string</Value>
            </Annotation>
        </annotations>
        <groupIdsDeleted>
            <groupId>guid</groupId>
            <groupId>guid</groupId>
        </groupIdsDeleted>
        <clientErrorData>string</clientErrorData>
    </contactInfo>
    <propertiesChanged>
        ContactPrimaryEmailType or
        ContactFirstName or ContactLastName or
        ContactQuickName or ContactBirthDate or
        ContactQuickOrder or
        ContactEmail or ContactPhone or ContactLocation or
        ContactWebSite
    </propertiesChanged>
    <fDeleted>boolean</fDeleted>
    <lastChange>dateTime</lastChange>
</Contact>
```

Contact.contactId: GUID used to identify this Contact. The contactId must be server generated. The Address Book does not support creating a new Contact with a partner supplied Contact Id.

Contact.propertiesChanged: Only used in ABContactUpdate. Set by the partner to indicate which fields should be updated. See ContactPropertyType. (ContactPrimaryEmailType, ContactFirstName, ContactLastName, ContactQuickName, ContactBirthDate, ContactQuickOrder, ContactEmail, ContactPhone, ContactLocation and ContactWebSite)

Contact.fDeleted: Used to indicate delta information. Set to true if this Contact has been deleted. Valid only when fDeltasOnly is true on the SOAP request.

Contact.lastChange: Format: GMT. ISO 8601 format. Purpose: Used by partners that keep a local cache of Groups and Contacts and need to synchronize with the Address Book.

ContactInfo.contactType: Regular or Me Contact indicator (ContactType enum)

ContactInfo.quickName: REQUIRED and UNIQUE. 60 char max. Will error if length exceeded. Quicknames are not case-sensitive. You cannot add two Contacts with Quicknames "aaa" and "AAA".

ContactInfo.firstName: 40 char max. If length is exceeded, value is truncated.

ContactInfo.lastName: 40 char max. If length is exceeded, value is truncated.

ContactInfo.passportName: Contains the Passport member name associated with the Contact. The Passport field is NOT REQUIRED. The Passport field, if provided in the Contact, MUST BE UNIQUE across all Contacts in that Address Book. The Passport field cannot be cleared if the isMessengerUser flag is set. 321 char max. (321 = 64 mailbox + "@" + 256 domain). Must be a valid SMTP address. The validation is less restrictive than the formal smtp specification. The validation will fail if there is: no @, @ is the first character, @ is the last character, no dot, dot is the last character, empty string, space is not the first or last character, or there is more than one @.

The Passport field must contain a VALID PASSPORT. The only exception is for Passports that have expired or have been deleted by the Passport holder.

Tombstone and Expiration

On tombstone (delete) of a Contact, the passportName must be cleared. If the TTL of the Passport has expired, and the Passport itself is no longer found or expired, then the isMessengerUser flag should be cleared at the same time the Passport Name entry is removed.

ContactInfo.puid: Only available through the IP Filtered Front end. The passportPuid and passportName can both be specified over the IP Filtered Front End. This will save a lookup on create and update operations.

ContactInfo.comment: Used to store comments or notes associated with the Contact in the Address Book. It is a user-editable field. Maximum limit of 1024. Can contain HTML and script – all tags/script should be stripped on input and output.

ContactInfo.isMobileVisible: Used to indicate whether or not the user has opted to view this contact on his/her mobile phone. Default is false.

ContactInfo.isMobileIMEnabled: Messenger: MOB

Mobile IM enabled? This is not "live". Messenger will propagate this (just like the friendly name) when both users are online. This value will be stored in the ABCH immediately upon change. 1=enable mobile IM. 0=no mobile IM.

ContactInfo.isMessengerUser: Can only be True if passportName has a value. True -- This Contact is a Messenger User. False -- This Contact is not a Messenger User.

This property cannot be set to true if the passportName or puid does not have a value.

ContactInfo.isFavorite: Indicates whether this Contact is in the Favorites category. Partners should set isFavorite to false for all new Contacts. On create, if not specified, isFavorite is set to false by the ABCH.

ContactInfo.isSmtp: True if this is an SMTP contact.

ContactInfo.spotWatchState: Messenger: SPOT Web Watch indicator (Smart Personal Object Technology). IM/pagers on web watch. Tri-state type. This value will be stored in the ABCH immediately upon change.

null= no web-watch
0= Web watch present but no messaging allowed
1= web-watch present and messaging allowed

ContactInfo.birthdate: ISO 8601 format. Must be passed in GMT. Will be returned in PST. The SQL Server smalldatetime can store dates/times between Jan 1, 1900 and June 6, 2079. If you supply a date and specify propertiesChanged=ContactBirthDate, the date MUST be within the SQL range - which is smaller than the .NET Framework range. See Dates for more information.

ContactInfo.quickOrder: The sequence of the Contact in the Quicklist of the partner UI. Not maintained by the Address Book. Field is nullable. It is 2 bytes so that it can be used to order up to 64k contacts.

ContactInfo.primaryEmailType: The email type that is the preferred smtp address to use when sending email. Must be set to: ContactEmailPersonal, ContactEmailBusiness, ContactEmailOther. May not be set to: ContactEmailMessenger.

ContactInfo.emails: See below.

ContactInfo.phones: See below.

ContactInfo.locations: See below.

ContactInfo.websites: See below.

ContactInfo.annotations: See below.

ContactInfo.groupsIds: List of Group Ids that this Contact is a member of.

ContactInfo.groupIdsDeleted: List of Group Ids that were deleted from the AB. Only valid if fDeltasOnly was set. This is essentially the list of tombstoned references for the requested Group.

ContactInfo.clientErrorData: Used by partners to identity the Contact that failed during a bulk operation. Only needed when specifying multiple Contacts in a request. This value is not stored in the ABCH. The value is only used during the request/response. The value is needed because a Contact Id will not have been generated and the Quickname information (which would otherwise be unique) - may be incorrect or invalid.

E-mail Address

ContactEmail.email: 321 char max.(321 = 64 mailbox + "@" + 256 domain)

Must be a valid SMTP address. The validation is less restrictive than the formal smtp specification. The validation will fail if there is: no @, @ is the first character, @ is the last character, no dot, dot is the last character, empty string, space is not the first or last character, more than one @.

ContactEmail.contactEmailType: ContactEmailPersonal, Business, Other, Messenger. Only one of each type is allowed in the Address Book.

Phone Number

ContactPhone.number: 50 char max. If length is exceeded, value is truncated. No other validation.

ContactPhone.contactPhoneType: ContactPhonePersonal, Business, Mobile, Page, Fax, Other, LivePersonal, LiveBusiness, LiveOther. Only one of each type is allowed in the Address Book.

Location

ContactLocation.name: 40 char max. If length is exceeded, value is truncated. No other validation. Company name.

ContactLocation.street: 256 char max. If length is exceeded, value is truncated. No other validation.

ContactLocation.city: 40 char max. If length is exceeded, value is truncated. No other validation.

ContactLocation.state: 40 char max. If length is exceeded, value is truncated. No other validation.

ContactLocation.postalCode: 40 char max. If length is exceeded, value is truncated. No other validation.

ContactLocation.country: 40 char max. If length is exceeded, value is truncated. No other validation.

ContactLocation.contactLocationType: ContactLocationPersonal, Business. Only one of each type is allowed in the Address Book.

Web Site

ContactWebSite.webURL: 2048 char max. If length is exceeded, value is truncated. No other validation.

ContactWebSite.contactWebSiteType: ContactWebSitePersonal, Business. Only one of each type is allowed in the Address Book.

A(iv). Me Contact

The Me Contact is a contact in the Address Book that holds information about the owner of the Address Book.

It is the same as a standard/regular contact but is differentiated by a new property; ContactType. If ContactInfo.contactType == ContactType.Me, the contact is the Me Contact.

The Me Contact is used to store information specific to the owner/user. For instance, certain annotations (MSNMobileEnabledAsked) are only specified on the Me Contact.

In the table below, ME indicates properties that apply to the ME Contact. Because the ContactInfo class is the same for both the ME and Regular Contacts, the properties are available to be set and read, however they only apply as indicated in the table.

Table:

Property Name	Me	Reg
ContactInfo.contactType	X	X
ContactInfo.isSmtp		X
ContactInfo.isFavorite		X
ContactInfo.isMobileIMEnabled	X	X
ContactInfo.spotWatchState	X	X

A(v). Annotations

Annotations are Name Value Pairs (NVPs) that can be associated with Groups and Contacts (or other objects in the future). All Annotations will be fully accessible by all partners.

There no limit to how many annotations can be added to an object. So to have some control over the growth of Annotations, we will be maintaining an AnnotationType list of supported Annotation Names. Only one instance of a given annotation name can be assigned to a given object.

The following properties will be associated with an Annotation:

Table:

```
<Contact>
  ...
      <contactInfo>
  ...
          <annotations>
              <Annotation>
                  <Name>string</Name>
                  <Value>string</Value>
              </Annotation>
          </annotations>
  ...
      </contactInfo>
  ...
</Contact>
```

Current Annotations: MSN.IM.MBEA: Messenger: MBEA

Mobile enabled bit (cache of Passport information). This field determines whether the user was prompted to enable Mobile IM. 0=no mobile device. 1=mobile enabled. 2=client must prompt.

This field is a cache of the Passport setting maintained by Messenger. (Note: These names are stored in the ABCH only once, and only a reference to the annotation type is stored with the value.)

MSN.IM.GTC: Messenger: GTC: 1=prompt, ask permission before reverse. 0=no prompt, permission not needed.

MSN.IM.BLP: Messenger: BLP

Permission setting for users not on Allow/Block. 1=allow list only. 0=block list excluded

B. SOAP Protocol

B(i). SOAP API Overview

Method Overview

Method and Description

ABAdd: Provision an Address Book (AB) in the service based on users PUID.

ABDelete: Delete an AB from the service. Deletes all Contacts and Groups associated with the AB.

ABUpdate: Update AB properties. Used to change the ownerEmail for an Address Book.

ABFind: Find an Address Book based on a users PUID. Use ABFind to retrieve AB properties.

ABContactAdd: Add one or more Contacts to the Address Book.

ABContactDelete: Delete one or more Contacts from the Address Book.

ABContactUpdate: Update Contact properties.

ABFindAll: Returns all Contacts, Groups and AB information in a single call.

ABFindByContacts: This method returns all of the Contacts with the given contact IDs as well as all of the Groups that contain those Contacts.

ABFindByGroups: Returns all of the contacts in the given groups and the groups themselves.

ABFindMessengerUsers: Returns all Contacts that have the isMessengerUser flag set - regardless of the Group membership. Also returns the "Me" contact and all Groups that contain any of the returned Contacts.

ABFindMeContact: Returns only the "Me" contact. Annotations are returned as well if relevant to the abView.

ABGroupAdd: Add a Group or set of Groups to the AB.

ABGroupDelete: Delete a Group or set of Groups from the AB.

ABGroupUpdate: Update Group properties. Used to change the name of the Group.

ABGroupFind: Find all Groups in the AB. This returns just the Group information and not the Contacts in the Group.

ABGroupContactAdd: Add a new Contact or existing Contact to a Group or set of Groups. Includes ability to merge Contact information.

ABGroupContactDelete: Delete a Contact from a Group or set of Groups. If the Contact is not contained in any other Groups after the requested delete, then the Contact will be deleted from the Address Book.

ABAllowListSet: Used to set the parental control allow list.

ABAllowListGet: Used to retrieve the parental control allow list.

The lastChange date is not returned on Add/Update/Delete operations. It is only returned on Find operations.

B(ii). HTTP Headers

A sample set of HTTP headers is included below.

The SOAPAction header is required for each call and includes the method name.

The Content-Type should include the charset qualifier. This is required by SOAP (see attached thread in this section). For example,

content type: text/xml; charset="utf-8"

Table: Sample HTTP header for SOAP request:

```
POST http://contacts.msn.com/ABService/ABService.asmx
HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web
Services Client Protocol 1.0.3705.0)
Content-Type: text/xml; charset=utf-8
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABAdd"
Content-Length: 439
Expect: 100-continue
Proxy-Connection: Keep-Alive
Host: contacts.msn.com
```

B(iii). SOAP Header

Each method call to the Address Book will be required to have additional properties passed in the header.

Table:

```
<soap:Header>
  <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
      <ApplicationId>guid</ApplicationId>
  </ABApplicationHeader>
  <ABAuthHeader
  xmlns="http://contacts.msn.com/webservices/AddressBook">
      <ManagedGroupRequest>boolean</ManagedGroupRequest>
  </ABAuthHeader>
</soap:Header>
```

ABAuthHeader.ManagedGroupRequest: If this SOAP request is a read, write, or provision request by the parent to a managed child account, this flag must be set to true. Note: The parent account has full

access to the childs account (managed account) address book. This flag is required as an optimization for the ABCH frontend. Special handling of this flag is discussed below.

Passport TimeWindow: This value can be set in the web.config file.

B(iii)a. Application Header

Used to identify the application calling the SOAP method.
This header is REQUIRED on calls.

ABApplicationHeader.ApplicationId: GUID to identify the Address Book partners.

B(iv). Globalization

All SOAP requests and responses pass strings in UTF-8 format.

There will not be an LCID stored for the Address Book itself or individual Contacts/Groups within the AB.

B(v). SOAP Method Bulk Limits

Many of the methods allow for multiple Contacts or Groups to be supplied or identified with a given request. There are limits to the actual number of Contacts or Groups that can be given in a request.

When Adding, Updating, or Deleting:

Maximum number of Contacts in a single SOAP request: 30
Maximum number of Groups in a single SOAP request: 30

Minimum number of Contacts in a single SOAP request: 1
Minimum number of Groups in a single SOAP request: 1

Note that the number of Contacts that may be returned during a Find is not restricted.

A minimum number of Contacts/Groups is imposed to insure NOP methods are not being produced by our partners. This will help improve performance in the service.

The maximum limit stems from the SQL limit for varchars being 8000 chars or less. The ABCH has to convert the list of guids into a string varchar. Because of this, and to be consistent across all our APIs, we need to keep a maximum limit for groupids, grouptypes, contactids in one bulk call for v1 (for add / update / delete).

B(vi). Usage of Passport PUID

B(vi)a. Passport Authed Front End

For the Passport Authed FE , PUIDs will not be passed in the clear.

The ABAdd()/ABFind() methods will allow a for a zero filled PUID to be passed. A zero filled PUID indicates to the Address Book to use cookies in the HTTP header to retrieve the PUID.

The AB Id returned from the ABAdd()/ABFind() will also be a zero filled identity. When this identity is passed to all other SOAP requests, the Address Book will be identified using the Authenication nodes in the SOAP header.

In the case of Parental Control related calls, the authentication header will contain the cookies of the parent. The AB Id will have to be assigned the PUID by the client. This call must be made over the SSL connection to the SOAP methods - to protect the information.

B(vii). Handling lastChange

B(vii)a. lastChange Description (generic)

Format: GMT. ISO 8601 format.

Purpose: Used by partners that keep a local cache of Groups and Contacts and need to synchronize with the Address Book.

B(vii)b. AB.lastChange

Initial value: When the AB is created, the lastChange value is set to the creation date.

Criteria for change: Set to Group.lastChange or Contact.lastChange whenever those values change. Set whenever ABFind is called. No more than once every 30 days. See notes below.

B(vii)c. Group.lastChange

Initial value: When the Group is created, the lastChange value is set to the creation date.

Criteria for change: Add an existing Contact, or new Contact, to the Group. Delete a Contact from the Group. Change any property on the Group itself (e.g. Group name).

Example of changes to the Group that will not affect this value:

- Any properties changed on Contacts does not affect the Group timestamp.

B(vii)d. Contact.lastChange

Initial value: When the Contact is created, the lastChange value is set to the creation date.

Criteria for change: Changing any property on the Contact itself. Tombstone operations: Set to the date Contact.fDeleted was set to true (delete) or false (resurrection).

Example of changes to the Contact that will not affect this value: Changing the Contact's Group membership in any Group (adding to a Group, deleting from a Group). Note there is a separate timestamp kept in the Address Book for each Group membership. That is the timestamp the Address Book also uses when returning changes on Groups.

When passing the lastChange through raw XML, lastChange can't be null or blank, but the whole element can be omitted (if you don't care) e.g. <lastChange/> fails, as does <lastChange></lastChange> and <lastChange>0</lastChange> If you want to pass a value and not omit, you can pass the value equivalent to System.DateTime.MinValue.

B(viii). Generating quickName

The quickname is generated by the Address Book.

B(ix). Dates

Partners will have to validate that if user types a date - month day year must be filled in, and the date must be between Jan 1, 1900 and June 6, 2079. A null date is equivalent to that date not being entered by the user. Here's more info about the date types:

B(ix)a. Timezones

All dates must be passed as GMT. When the date is returned to the caller, the date will be in PST. The caller will have to convert the date back from PST to GMT to get the same value that was passed in.

B(ix)b. "NULL" Dates

If you set the birthdate null <birthdate/>, the .NET Framework converts that to 01/01/0001, which we recognize and store in SQL as a null.

When you retrieve a null birthdate, you get 01/01/0001 - it looks like this in the XML:

birthdate>0001-01-01T00:00:00.0000000</birthdate>

B(ix)c. Date Ranges

The .NET Framework DateTime value type represents dates and times with values ranging from 12:00:00 midnight, January 1, 0001 C.E. (Common Era) to 11:59:59 P.M., December 31, 9999 C.E.

The SQL Server smalldatetime can store dates/times between Jan 1, 1900 and June 6, 2079.

If you supply a date and specify propertiesChanged=ContactBirthDate, the date MUST be within the SQL range - which is smaller than the .NET Framework range.

B(x). Return Id on Conflict

When adding a Contact or Group, if there is a conflict with the Quickname, .NET Passport field, or Group Name, the ABCH will return the Id of the conflicting object.

To avoid Quickname conflicts, use fGenerateMissingQuickname (or leave the Quickname blank) to auto-generate a Quickname.

SOAP Faults: GroupAlreadyExists, ContactAlreadyExists

(omitted some irrelevant SOAP fault elements):

Table:

```
<soap:Fault>
  <detail>
       <errorcode
xmlns="http://contacts.msn.com/webservices/AddressBook">
       GroupAlreadyExists
       </errorcode>
       <additionalDetails>
            <conflictObjectId>
                 0713AEBB-2632-451D-A8B7-61AAF9CEE476
            </conflictObjectId>
       </additionalDetails>
  </detail>
</soap:Fault>
```

The ID can be parsed by using XPath "additionalDetails/conflictObjectId" on the caller side. Returning this piece of information will make the error handling for ContactAlreadyExists/GroupAlreadyExists error more efficient.

The following APIs can throw SoapException with this information:

- ABGroupAdd, when another group with the same GroupName already exists in ABCH;
- ABGroupUpdate, same as above
- ABContactAdd, when another contact with the same QuickName or same PassportId already exists in ABCH;
- ABContactUpdate, same
- ABGroupContactAdd, same

C. Methods

C(i). ABAdd

Provision a new Address Book in the service based on users PUID.

C(i)a. ABAdd

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public Guid ABAdd(      ABInfo abInfo)
```

Parameters

[hdr] Auth Header and Application Header

If the caller (the PUID identified in the header) does not match the Owner, and the Owner is a managed PUID, the AB will be provisioned if the caller is a "parent" of the managed PUID. This will allow a parent to establish an AB for a child, and pre-populate the allow list using ABAllowListSet.

See the SOAP Header section for more information.

[in] abInfo

Properties for the Address Book.

.name	Do not use - pass null string or empty element.
.ownerPuid	REQUIRED: PUID for the user that owns the Address Book. For clients using the Passport authed FE, the ownerPuid should be set to zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request. In Parental Control requests, the ABAdd may be the parent provisioning an AB for a child PUID. In this case, the client must populate ownerPuid with the childs PUID, and MAKE THE CALL OVER SSL to protect the privacy of the data. In this case, the ManagedGroupRequest flag must be set to true in the header as well.
.ownerEmail	Email address of the owner of the Address Book.
.fDefault	Reserved for a future version when multiple Address Books may exist for a given PUID. Currently, always pass true in ABAdd.

[return] Guid

Guid for the Address Book.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://contacts.msn.com/webservices/AddressBook/ABAdd"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
```

```
                <ManagedGroupRequest>boolean</ManagedGroupRequest>
            </ABAuthHeader>
        </soap:Header>
        <soap:Body>
            <ABAdd
xmlns="http://contacts.msn.com/webservices/AddressBook">
                <abInfo>
                    <name>string</name>
                    <ownerPuid>long</ownerPuid>
                    <ownerEmail>string</ownerEmail>
                    <fDefault>boolean</fDefault>

        <isMessengerMigrated>boolean</isMessengerMigrated>
                </abInfo>
            </ABAdd>
        </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
            <ABAddResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
                <ABAddResult>guid</ABAddResult>
            </ABAddResponse>
        </soap:Body>
</soap:Envelope>
```

C(i)b. Additional Notes

This method should be invoked when an "AB Does Not Exist" error is returned from one of the other methods.

In parental control scenarios, the parent may need to provision the Address Book for the child. In this case: The <ApplicationId> must be the ID for child's client application. The cookies provided in the HTTP header are the cookies of the parent. The <ManagedGroupRequest> flag MUST be set to true. The <ownerPUID> is the PUID of the child. The <ownerEmail> is the child's email. ABAdd MUST be made over SSL as the PUID of the child Address Book is passed in the clear.

C(ii). ABDelete

Delete an Address Book from the service. Deletes all Contacts and Groups associated with the Address Book.

C(ii)a. ABDelete

[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public void ABDelete(Guid abId)

Parameters

[hdr] Auth Header and Application Header: *See the SOAP Header section for more information.*

[in] abId: Identifies the Address Book to delete. For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABDelete
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
          </ABDelete>
     </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABDeleteResponse
xmlns="http://contacts.msn.com/webservices/AddressBook" />
     </soap:Body>
</soap:Envelope>
```

C(iii). ABUpdate

Update Address Book properties.

C(iii)a. ABUpdate

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public void ABUpdate(AB ab)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Address Book section at the beginning of the appendix.

[hdr] Auth Header and Application Header: See the SOAP Header section for more information.

[in] ab: Identifies the Address Book to update and the new properties.

.abId: Must be set to the identity of the Address Book to change. For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

.abInfo.name: Friendly name of the Address Book.

.abInfo.ownerPuid
.abInfo.ownerEmail
.abInfo.fDefault: Cannot be updated.

.propertiesChanged: Set by the caller to indicate which fields should be updated. See ABPropertyType. ABName

.createDate
.lastChange: Cannot be updated.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABUpdate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABUpdate
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ab>
                    <abId>guid</abId>
                    <abInfo>
                         <name>string</name>
                         <ownerPuid>long</ownerPuid>
                         <ownerEmail>string</ownerEmail>
                         <fDefault>boolean</fDefault>

     <isMessengerMigrated>boolean</isMessengerMigrated>
                    </abInfo>
                    <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                    <createDate>dateTime</createDate>
                    <lastChange>dateTime</lastChange>
               </ab>
          </ABUpdate>
     </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABUpdateResponse
xmlns="http://contacts.msn.com/webservices/AddressBook" />
     </soap:Body>
</soap:Envelope>
```

C(iv). ABFind

Find an Address Book from the service.

C(iv)a. ABFind

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public AB[] ABFind(Int64 ownerPuid)
```

Parameters

[hdr] Auth Header and Application Header: See the SOAP Header section for more information.

[in] ownerPuid: Identifies the Owner of the AB. For clients using the Passport authed FE, the ownerPuid should be set to zero. This will cause the Address Book to get the PUID from HTTP header. In Parental Control requests, the ABFind may be the parent looking for the AB for a child PUID. In this case, the client must populate ownerPuid with the childs PUID, and MAKE THE CALL OVER SSL to protect the privacy of the data. In this case, the ManagedGroupRequest flag must be set to true in the header as well. PLEASE READ THE SECTION ON ownerPuid IN THE ABAdd GENERAL DESCRIPTION.

[return] AB: Returns the Address Book associated with the PUID. Currently, only one AB is supported per PUID. Only one AB will be returned if exists. Returns null if the Address Book does not exist (is not provisioned). An exception is not returned in this case.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://contacts.msn.com/webservices/AddressBook/ABFind"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
```

```
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABFind
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ownerPuid>long</ownerPuid>
          </ABFind>
     </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABFindResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ABFindResult>
                    <AB>
                         <abId>guid</abId>
                         <abInfo>
                              <name>string</name>
                              <ownerPuid>long</ownerPuid>
                              <ownerEmail>string</ownerEmail>
                              <fDefault>boolean</fDefault>

     <isMessengerMigrated>boolean</isMessengerMigrated>
                         </abInfo>
                         <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                         <createDate>dateTime</createDate>
                         <lastChange>dateTime</lastChange>
                    </AB>
                    <AB>
                         <abId>guid</abId>
                         <abInfo>
                              <name>string</name>
```

```
                                <ownerPuid>long</ownerPuid>
                                <ownerEmail>string</ownerEmail>
                                <fDefault>boolean</fDefault>

      <isMessengerMigrated>boolean</isMessengerMigrated>
                            </abInfo>
                            <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                            <createDate>dateTime</createDate>
                            <lastChange>dateTime</lastChange>
                        </AB>
                    </ABFindResult>
                </ABFindResponse>
      </soap:Body>
</soap:Envelope>
```

C(iv)b. Additional Notes

ABFind will have the side effect of writing the lastChange in the AB.

C(v). ABContactAdd

This method is used to add one or more Contacts to the Address Book.

C(v)a. ABContactAdd

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public Guid[] ABContactAdd(Guid abId, Contact[] contacts)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the appendix.

[hdr] Auth Header and Application Header: See the SOAP Header section for more information.

[in] abId: Adds the Contacts to Groups in this Address Book. For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] contacts: One or more Contacts to add. A null array entry will give a NullArgument exception.

.contactId: Should be null.

.contactInfo.quickName: . The Contact quickName is a required and unique field in the ABCH. If you do not supply a Quickname, a Quickname will be generated by default. 1) By default, we will not try to generate missing quickname or rename any input quickname on conflict. 2) If the first contact object in the input array has a null or

empty quickname, we will auto-generate missing quicknames and modify input quicknames on conflict for all contacts in the input array.

.contactInfo.firstName/.lastName
.contactInfo.birthdate
.contactInfo.quickOrder
.contactInfo.primaryEmailType: See Contact table for more information.

.contactInfo.emails
.contactInfo.phones
.contactInfo.locations
.contactInfo.websites

Note that within each array, only one occurrence of each subtype is allowed. For example, within the emails array, an email of type ContactEmailPersonal can only appear once in the array. See Contact table for more information.

.contactInfo.groupIds
.contactInfo.groupIdsDeleted: Not used during Add. Ignore.

.contactInfo.clientErrorData: Include a client determined value for this field when Adding multiple Contacts. This value will be returned as the identity of the Contact should that Contact fail the Add for any reason.

.propertiesChanged: ContactPrimaryEmailType, ContactFirstName, ContactLastName, ContactQuickName, ContactBirthDate, ContactQuickOrder, ContactEmail, ContactPhone, ContactLocation, and ContactWebSite.

[return] Guid: The list of identities for the specified Contacts is returned.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABContactAdd"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
      <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
          <ApplicationId>guid</ApplicationId>
          <IsMigration>boolean</IsMigration>
      </ABApplicationHeader>
```

```
        <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
             <ManagedGroupRequest>boolean</ManagedGroupRequest>
        </ABAuthHeader>
   </soap:Header>
   <soap:Body>
        <ABContactAdd
xmlns="http://contacts.msn.com/webservices/AddressBook">
             <abId>guid</abId>
             <contacts>
                  <Contact>
                       <contactId>guid</contactId>
                       <contactInfo>
                            <contactType>Regular or
Me</contactType>
                            <quickName>string</quickName>
                            <firstName>string</firstName>
                            <lastName>string</lastName>
                            <passportName>string</passportName>
                            <puid>long</puid>
                            <comment>string</comment>

   <isMobileVisible>boolean</isMobileVisible>

   <isMobileIMEnabled>boolean</isMobileIMEnabled>

   <isMessengerUser>boolean</isMessengerUser>
                            <isFavorite>boolean</isFavorite>
                            <isSmtp>boolean</isSmtp>
                            <spotWatchState>
                                 NoDevice or
                                 NoMessaging or
                                 MessagingEnabled
                            </spotWatchState>
                            <birthdate>dateTime</birthdate>
                            <quickOrder>short</quickOrder>
                            <primaryEmailType>
                                 ContactEmailPersonal or
                                 ContactEmailBusiness or
                                 ContactEmailOther or
                                 ContactEmailMessenger
                            </primaryEmailType>
                            <emails>
                                 <ContactEmail>
                                           <contactEmailType>

   ContactEmailPersonal
        ContactEmailBusiness
             ContactEmailOther or
```

```
   ContactEmailMessenger
                                        </contactEmailType>
                                   <email>string</email>
                              </ContactEmail>
                         </emails>
                         <phones>
                              <ContactPhone>
                                   <contactPhoneType>
                                        ContactPhonePersonal or
                                        ContactPhoneBusiness or
                                        ContactPhoneMobile or
                                        ContactPhonePager or
                                        ContactPhoneFax or
                                        ContactPhoneOther
                                   </contactPhoneType>
                                   <number>string</number>
                              </ContactPhone>
                         </phones>
                         <locations>
                              <ContactLocation>
                                   <contactLocationType>
                                        ContactLocationPersonal
or
                                        ContactLocationBusiness
                                   </contactLocationType>
                                   <name>string</name>
                                   <street>string</street>
                                   <city>string</city>
                                   <country>string</country>

   <postalCode>string</postalCode>
                               </ContactLocation>
                         </locations>
                         <webSites>
                              <ContactWebSite>
                                   <contactWebSiteType>
                                        ContactWebsitePersonal
or
                                        ContactWebsiteBusiness
                                   </contactWebSiteType>
                                   <webURL>string</webURL>
                              </ContactWebSite>
                         </webSites>
                         <annotations>
                              <Annotation>
                                   <Name>
                                        MSN.IM.GTC or
                                        MSN.IM.BLP or
```

```
                                        MSN.IM.MBEA
                                    </Name>
                                    <Value>string</Value>
                                </Annotation>
                            </annotations>
                            <groupIds xsi:nil="true" />
                            <groupIdsDeleted xsi:nil="true" />

  <clientErrorData>string</clientErrorData>
                        </contactInfo>
                        <propertiesChanged>
                            ContactPrimaryEmailType or
                            ContactFirstName or
                            ContactLastName or
                            ContactQuickName or
                            ContactBirthDate or
                            ContactQuickOrder or
                            ContactEmail or
                            ContactPhone or
                            ContactLocation or
                            ContactWebSite or
                            Annotation or
                            Passport or
                            Comment or
                            IsMobileVisible or
                            IsMobileIMEnabled or
                            IsMessengerUser or
                            IsFavorite or
                            IsSmtp or
                            SpotWatchState or
                        </propertiesChanged>
                        <fDeleted>boolean</fDeleted>
                        <lastChange>dateTime</lastChange>
                    </Contact>
                </contacts>
            </ABContactAdd>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
```

```
        <ABContactAddResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
            <ABContactAddResult>
                <guid>guid</guid>
                <guid>guid</guid>
            </ABContactAddResult>
        </ABContactAddResponse>
    </soap:Body>
</soap:Envelope>
```

C(v)b. Special Notes

The Contacts are NOT added to the MSNAB Group using this method.

There is no umbrella transaction over bulk Contact Add operations. If one of the Contacts fails to add, all Adds prior to that failure MAY be committed. The client must perform an fDeltasOnly synchronization to determine which Contacts had committed. Any Contact Adds after the failure are discarded and must be retried by the client.

C(v)c. Add Contact Parental Controls

See the section on Parental Controls for more information.

Through the Passport auth'ed Front End, Managed accounts will not be allowed to Add a Messenger User.

C(v)d. Return Id on Conflict

When adding a Contact whose quickname or passportName conflicts with another Contact, the Id of the conflicting Contact will be returned.

See the section on Return Id on Conflict for more information.

C(v)e. contactType and isSmtp defaults

If contactType is NOT specified, the contactType will default to Regular.

If isSmtp is NOT specified, isSmtp is set to false.

C(v)f. isSmtp MUST be false

ABGroupContactAdd must be used to add a Contact where isSmtp=true. You cannot add an SMTP Contact through ABContactAdd.

C(vi). ABContactDelete

Delete a Contact from the Address Book.

C(vi)a. ABContactDelete

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public void ABContactDelete(Guid abId, Contact[] contacts)
```

Parameters

[hdr] Auth Header and Application Header: See the SOAP Header section for more information.

[in] abId: Identifies the Address Book that contains the Group. For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] contacts: Identifies the Contact to delete from the Group. Contact can be identified by Id only.

.contactId: Contact ID for the Contact to be removed from the specified Groups.

.contactInfo: MUST be null.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABContactDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABContactDelete
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <contacts>
                    <Contact>
                         <contactId>guid</contactId>
                         <contactInfo/>
                         <propertiesChanged>
```

```
                                ContactPrimaryEmailType or
                                ContactFirstName or
                                ContactLastName or
                                ContactQuickName or
                                ContactBirthDate or
                                ContactQuickOrder or
                                ContactEmail or
                                ContactPhone or
                                ContactLocation or
                                ContactWebSite or
                                Annotation or
                                Passport or
                                Comment or
                                IsMobileVisible or
                                IsMobileIMEnabled or
                                IsMessengerUser or
                                IsFavorite or
                                IsSmtp or
                                SpotWatchState or
                            </propertiesChanged>
                            <fDeleted>boolean</fDeleted>
                            <lastChange>dateTime</lastChange>
                        </Contact>
                    </contacts>
                </ABContactDelete>
            </soap:Body>
        </soap:Envelope>

        HTTP/1.1 200 OK
        Content-Type: text/xml; charset=utf-8
        Content-Length: length

        <?xml version="1.0" encoding="utf-8"?>
        <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
            <soap:Body>
                <ABContactDeleteResponse
        xmlns="http://contacts.msn.com/webservices/AddressBook" />
            </soap:Body>
        </soap:Envelope>
```

C(vi)b. Additional Notes

If this Contact is a member of one or more Groups, the Contact is deleted from those Groups as well. There is no mechanism to remove a Contact from the Address Book without removing a Contact from all the Groups it belongs to.

Bulk Contact deletes are transacted. If one Contact fails to delete, all the deletions are rolled back.

C(vii). ABContactFind

Find zero or more Contacts in an Address Book based on the search criteria.

C(vii)a. ABContactFind

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public ContactFindResult ABContactFind(
        Guid abId,
        ContactFilter contactFilter
)
```

Parameters

Information specific to ABContactFind is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the appendix.

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Find Contacts in this Address Book.

For clients using the Passport authed FE, the abId MUST be zero. This will cause the Clearinghouse to get the PUID from the authentication header in the SOAP request.

[in] contactFilter

Criteria for the search. See the ContactFilter table for more information.

If the contactFilter is null, you will receive an error.

If neither groupFilter nor contactIds are provided, all the Contacts will be returned (pending further filtering by lastChange/fDeltasOnly).

If a Guid provided in a groupIds or contactIds does not exist, no error will result. The result set may in fact be empty if only non-existing groupsIds are specified.

If both a groupFilter and contactIds are specified, the union of the result will be returned. Contacts will not be returned twice.

.groupFilter.groupIds

NOT supported.

.groupFilter.groupTypes

Return the Contact members of these Groups.

Note you cannot just specify MSNPUBLIC. You must specify MSNAB or MSNBUDDY (or both) with MSNPUBLIC.

For groupTypes=MSNAB, ALL the Rich Contacts in the Address Book are returned. The actual membership of MSNAB is not used in R9.

.contactIds

Specify a set of individual Contacts to find.

Cannot be used with ContactInfoMessenger.

Searching on contactIds and groupTypes at the same time is not supported.

.lastChange

If lastChange is specified:

- The lastChange value must be from a previous call to ABContactFind, or be DateTime.MinValue (or in XML just leave the lastChange element out).
- If fDeltasOnly is set to true, all Contacts changed since the lastChange date will be returned.
- And fDeltasOnly is set to false, all the Contacts will be returned if *any* of the Contacts have changed since the lastChanged date.

See notes on lastChange below. Also in the section Handling lastChange.

.contactInfoView

Note: Value types will be returned with the correct value even if not specified in the view. This is a bug fix over R8. Value types include dates, boolean and integers.

ContactInfoHotmail
ContactInfoMessenger
ContactInfoAll

.fDeltasOnly

If true, return on the changes since the lastChanged date.

[return] ContactFindResult

.groups Group information for all Groups referenced by the Contacts returned in the call. Empty groups based on the Group filter you provide will also be returned. Empty groups are not returned if contactIds are supplied.

ABContactFind returns the list of Groups (including empty Groups) before it returns the list of Contacts. This has 2 big benefits:
- The complete Group information is not repeated inside each Contact. Instead, ABContactFind just lists the groupIds for Groups that the Contact belongs to. Less data flowing across the wire.

.contacts The Contacts in the result list. There is no limit to the number of Contacts that may be returned in a request.

.ablastChange Current timestamp in the AB.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.msn.com/webservices/AddressBook/ABContactFind"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <ABApplicationHeader xmlns="http://www.msn.com/webservices/AddressBook">
      <ApplicationId>guid</ApplicationId>
      <IsMigration>boolean</IsMigration>
    </ABApplicationHeader>
    <ABAuthHeader xmlns="http://www.msn.com/webservices/AddressBook">
      <ManagedGroupRequest>boolean</ManagedGroupRequest>
    </ABAuthHeader>
  </soap:Header>
  <soap:Body>
    <ABContactFind xmlns="http://www.msn.com/webservices/AddressBook">
      <abId>guid</abId>
      <contactFilter>
        <groupFilter>
          <groupIds>
            <guid>guid</guid>
            <guid>guid</guid>
          </groupIds>
          <groupTypes>
            <guid>guid</guid>
            <guid>guid</guid>
          </groupTypes>
        </groupFilter>
        <contactIds>
          <guid>guid</guid>
          <guid>guid</guid>
        </contactIds>
        <lastChange>dateTime</lastChange>
```

```
        <contactInfoView>ContactInfoHotmail or ContactInfoMessenger or
ContactInfoAll</contactInfoView>
        <fDeltasOnly>boolean</fDeltasOnly>
      </contactFilter>
    </ABContactFind>
  </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ABContactFindResponse xmlns="http://www.msn.com/webservices/AddressBook">
      <ABContactFindResult>
        <groups>
          <Group>
            <groupId>guid</groupId>
            <groupInfo xsi:nil="true" />
            <propertiesChanged>GroupName or Annotation or
QuickOrder</propertiesChanged>
            <fDeleted>boolean</fDeleted>
            <lastChange>dateTime</lastChange>
          </Group>
          <Group>
            <groupId>guid</groupId>
            <groupInfo xsi:nil="true" />
            <propertiesChanged>GroupName or Annotation or
QuickOrder</propertiesChanged>
            <fDeleted>boolean</fDeleted>
            <lastChange>dateTime</lastChange>
          </Group>
        </groups>
        <contacts>
          <Contact>
            <contactId>guid</contactId>
            <contactInfo xsi:nil="true" />
            <propertiesChanged>ContactPrimaryEmailType or ContactFirstName or
ContactLastName or ContactQuickName or ContactBirthDate or ContactQuickOrder or
ContactEmail or ContactPhone or ContactLocation or ContactWebSite or Annotation or
Passport or Comment or IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState or DisplayName</propertiesChanged>
            <fDeleted>boolean</fDeleted>
            <lastChange>dateTime</lastChange>
          </Contact>
          <Contact>
            <contactId>guid</contactId>
            <contactInfo xsi:nil="true" />
            <propertiesChanged>ContactPrimaryEmailType or ContactFirstName or
ContactLastName or ContactQuickName or ContactBirthDate or ContactQuickOrder or
ContactEmail or ContactPhone or ContactLocation or ContactWebSite or Annotation or
Passport or Comment or IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState or DisplayName</propertiesChanged>
```

```
            <fDeleted>boolean</fDeleted>
            <lastChange>dateTime</lastChange>
          </Contact>
        </contacts>
        <ablastChange>dateTime</ablastChange>
      </ABContactFindResult>
    </ABContactFindResponse>
  </soap:Body>
</soap:Envelope>
```

C(vii)b. Additional Notes

The search criteria is limited. You can find a specific Contact or Contacts, or you can Find the membership of a Group. You cannot find a Contact by Quickname or SMTP address in this release.

Unless otherwise specified in the filter, this methods searches on the list of all Contacts contained in the AB, regardless of which Groups the Contacts have joined.

The Groups that the Contact belongs to will be returned with the Contact information.

C(vii)c. lastChange

The following applies when fDeltasOnly is true:

When ContactFilter.lastChange is used with ABContactFind, the value must match a ContactFindResult.ablastChange returned by a previous call to ABContactFind, or be DateTime.MinValue (or in XML just leave the lastChange element out). If the value is later than the last ablastChange, or earlier than the Address Book creation, an error will result:

InvalidSyncTimeStamp - Indicates a ContactFilter.lastChange date later than the last available lastChange date. This should never be returned if clients always use timestamps (ablastChange) returned from ABContactFind

FullResyncRequired - Indicates a ContactFilter.lastChange date earlier than the creation date for the AB or client is out of sync by more than 90 days.

Note that delta sync on ABContactFind will be supported only for the below combination of parameters (which should match the Hotmail delta sync calls to ABCH)

More information on lastChange can be found in Handling lastChange.

C(vii)d. A query on the MSNAB Group returns ALL Rich Contacts (Everyone)

For groupTypes=MSNAB, ALL the Rich Contacts in the Address Book are returned. The actual membership of MSNAB is not used in R9.

C(vii)e. contactFilter changes

Searching by groupFilter.groupIds is NOT supported.

Searching by groupFilter.groupTypes is supported. Note you cannot just specify MSNPUBLIC. You must specify MSNAB or MSNBUDDY (or both) with MSNPUBLIC.

Searching by contactIds is supported. Note searching on contactIds and groupTypes at the same time is not supported.

contactInfoView is supported. Note: Value types will be returned with the correct value even if not specified in the view. Value types include dates, boolean and integers.

C(vii)f. New Properties

See the section on New Contact Properties for more information on additional fields available to be specified.

C(viii). ABContactUpdate

Update Contact information. Multiple Contacts are allowed in one request.

C(viii)a. ABContactUpdate

[SoapHeader("m_abAuthHeader", Required=false)]

[SoapHeader("m_abAppHeader", Required=true)]
public void ABContactUpdate(Guid abId, Contact[] contacts)

Parameters

[hdr] Auth Header and Application Header: See the SOAP Header section for more information.

[in] abId: Target Address Book. For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] contacts: One or more Contacts to update. See Bulk Limits and the Contact table for more info. A null array entry will give a NullArgument exception. Information specific to ABContactUpdate is listed here. The rest of the information on the fields can be found in the Contact section at the beginning of the document.

.contact.contactId: The Contact ID is required and is the only mechanism to identify the Contact. Exception to this rule: When updating the ME Contact, the contactId should be null.

.contactInfo.contactType: Set to ContactType.Regular for a standard contact (e.g. not the ME Contact). Set to ContactType.Me to update the Me Contact. For more information on the ME Contact, see Me Contact.

.contactInfo.quickName: The Contact quickName is a required and unique field in the ABCH. It cannot be updated to null. You cannot update a Contact using the quickName as the key, you must supply the contactId. See the validation rules for a quickName in the Contact table.

.contactInfo.passportName: The MSN Member Name for this Contact. The passportName CANNOT be updated on the ME Contact. The ABCH uses the passportName from the ABInfo to populate this field. Please note special handling of contactInfo.puid, and how it affects this field.

.contactInfo.puid: Can only be updated over the IP Filtered (Partner) Front End. The puid CANNOT be updated on the ME Contact. The ABCH uses the

puid of the ABInfo passportName to populate this field. If this value is supplied, it overrides any value passed for the passportName, because the PUID always takes precedence. The PassportName will be updated through the Passport Name refresh mechanism in the ABCH.

.contactInfo.displayName: Messenger Display name.

.contactInfo.isMobileVisible
.contactInfo.isMobileIMEnabled:
See Contact properties for more information.

.contactInfo.isMessengerUser
.contactInfo.isFavorite:
Cannot be updated on the ME Contact.
See Contact properties for more information.

.contactInfo.isSmtp: Cannot be updated regardless of contactType.

.contactInfo.spotWatchState: See Contact properties for more information.

.contactInfo.firstName/.lastName
.contactInfo.birthdate
.contactInfo.quickOrder
.contactInfo.primaryEmailType: See the Contact table for more information.

.contactInfo.emails
.contactInfo.phones
.contactInfo.locations
.contactInfo.websites: To update any entry, propertiesChanged must contain the general type (e.g. phone, location, etc). To update or add a single entry, include that entry for that subtype with the new value (e.g. ContactEmailOther, "other@msn.com"). You do not need to supply all the emails again, just the one that changed. When updating location, fill in all the fields (street, city, etc) for that entry. To delete a single entry, simply include a that entry for that subtype with a null value. (e.g. ContactEmailOther, null). Note that within each array, only one occurrence of each subtype is allowed. For example, within the emails array, an email of type ContactEmailPersonal can only appear once in the array.

.contactInfo.groupIds/.groupIdsDeleted: The group membership cannot be changed through update. Use ABGroupContactAdd/Delete instead.

.contactInfo.clientErrorData: Include a client determined value for this field when updating multiple Contacts. This value will be returned as the identity of the Contact should that Contact fail the update for any reason.

.propertiesChanged: Required. Only used in ABContactUpdate. Set by the partner to indicate which fields should be updated. See ContactPropertyType
(ContactPrimaryEmailType,
ContactFirstName,
ContactLastName,

ContactQuickName,
ContactBirthDate,
ContactQuickOrder,
ContactEmail,
ContactPhone,
ContactLocation,
ContactWebSite)

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABContactUpdate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
      <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
          <ApplicationId>guid</ApplicationId>
          <IsMigration>boolean</IsMigration>
      </ABApplicationHeader>
      <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
          <ManagedGroupRequest>boolean</ManagedGroupRequest>
      </ABAuthHeader>
  </soap:Header>
  <soap:Body>
      <ABContactUpdate
xmlns="http://contacts.msn.com/webservices/AddressBook">
          <abId>guid</abId>
          <contacts>
              <Contact>
                  <contactId>guid</contactId>
                  <contactInfo>
                      <contactType>Regular or
Me</contactType>
                      <quickName>string</quickName>
                      <firstName>string</firstName>
                      <lastName>string</lastName>
                      <passportName>string</passportName>
                      <puid>long</puid>
                      <comment>string</comment>
```

```
<isMobileVisible>boolean</isMobileVisible>

<isMobileIMEnabled>boolean</isMobileIMEnabled>

<isMessengerUser>boolean</isMessengerUser>
                        <isFavorite>boolean</isFavorite>
                        <isSmtp>boolean</isSmtp>
                        <spotWatchState>
                             NoDevice or
                             NoMessaging or
                             MessagingEnabled
                        </spotWatchState>
                        <birthdate>dateTime</birthdate>
                        <quickOrder>short</quickOrder>
                        <primaryEmailType>
                             ContactEmailPersonal or
                             ContactEmailBusiness or
                             ContactEmailOther or
                             ContactEmailMessenger
                        </primaryEmailType>
                        <emails>
                             <ContactEmail>
                                  <contactEmailType>
                                       ContactEmailPersonal or
                                       ContactEmailBusiness or
                                       ContactEmailOther or
                                       ContactEmailMessenger
                                  </contactEmailType>
                                  <email>string</email>
                             </ContactEmail>
                        </emails>
                        <phones>
                             <ContactPhone>
                                  <contactPhoneType>
                                       ContactPhonePersonal or
                                       ContactPhoneBusiness or
                                       ContactPhoneMobile or
                                       ContactPhonePager or
                                       ContactPhoneFax or
                                       ContactPhoneOther
                                  </contactPhoneType>
                                  <number>string</number>
                             </ContactPhone>
                        </phones>
                        <locations>
                             <ContactLocation>
                                  <contactLocationType>
```

```
                                        ContactLocationPersonal
or
                                        ContactLocationBusiness
                                    </contactLocationType>
                                    <name>string</name>
                                    <street>string</street>
                                    <city>string</city>
                                    <country>string</country>

  <postalCode>string</postalCode>
                                </ContactLocation>
                            </locations>
                            <webSites>
                                <ContactWebSite>
                                    <contactWebSiteType>
                                        ContactWebsitePersonal
or
                                        ContactWebsiteBusiness
                                    </contactWebSiteType>
                                    <webURL>string</webURL>
                                </ContactWebSite>
                            </webSites>
                            <annotations>
                                <Annotation>
                                    <Name>
                                    MSN.IM.GTC or
                                    MSN.IM.BLP or
                                    MSN.IM.MBEA
                                    </Name>
                                    <Value>string</Value>
                                </Annotation>
                            </annotations>
                            <groupIds xsi:nil="true" />
                            <groupIdsDeleted xsi:nil="true" />

  <clientErrorData>string</clientErrorData>
                        </contactInfo>
                        <propertiesChanged>
                            ContactPrimaryEmailType or
                            ContactFirstName or
                            ContactLastName or
                            ContactQuickName or
                            ContactBirthDate or
                            ContactQuickOrder or
                            ContactEmail or
                            ContactPhone or
                            ContactLocation or
                            ContactWebSite or
                            Annotation or
```

```
                                    Passport or
                                    Comment or
                                    IsMobileVisible or
                                    IsMobileIMEnabled or
                                    IsMessengerUser or
                                    IsFavorite or
                                    IsSmtp or
                                    SpotWatchState or
                                </propertiesChanged>
                                <fDeleted>boolean</fDeleted>
                                <lastChange>dateTime</lastChange>
                            </Contact>
                        </contacts>
                    </ABContactUpdate>
                </soap:Body>
            </soap:Envelope>

            HTTP/1.1 200 OK
            Content-Type: text/xml; charset=utf-8
            Content-Length: length

            <?xml version="1.0" encoding="utf-8"?>
            <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
            instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
                <soap:Body>
                    <ABContactUpdateResponse
            xmlns="http://contacts.msn.com/webservices/AddressBook" />
                </soap:Body>
            </soap:Envelope>
```

C(viii)b. Additional Notes

There is no umbrella transaction over bulk Contact Update operations. If one of the Contacts fails to update, all Updates prior to that failure MAY be committed. The client must perform an fDeltasOnly synchronization to determine which Contacts had committed. Any Contact updates after the failure are discarded and must be retried by the client.

C(viii)c. Changes to IsMessengerUser and passportName

Through update, the Messenger User information can be removed, or replaced by different IM address information. The Uses Messenger flag can change. The behavior is equivalent to add and delete semantics described in ABGroupContactAdd and ABGroupContactDelete above.

C(viii)d. The ME Contact

Restrictions on the ME Contact: Cannot be added to a Group, Cannot be set isFavorite, Cannot be set isMessengerUser.

C(viii)e. contactType

The Contact type CANNOT be updated! propertiesChanged will not be available for contactType

C(viii)f. Return Id on Conflict

When adding a Contact whose quickname or passportName conflicts with another Contact, the Id of the conflicting Contact will be returned. See the section on Return Id on Conflict for more information.

C(viii)g. New Properties

See the section on New Contact Properties for more information on additional fields available to be specified.

C(ix). ABFindAll

Returns all Contacts, Groups and AB information in a single call.

C(ix)a. ABFindAll

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public FindResult ABFindAll(Guid abId, ABView abView, bool deltasOnly, DateTime
lastChange)
```

Parameters

[hdr] Auth Header and Application Header: See the SOAP Header section in the Contacts API spec for more information.

[in] abId: Find Contacts in this Address Book. For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] abView: See the section on ABView in ABFindAll for more information.

[in] deltasOnly: Should differences since lastChange be sent? If deltasOnly is true, and the lastChange is set in the filter, only the Contacts that have changed since the lastChange date will be returned to the caller. A Contact may have changed because either the properties have changed, or the membership in the Group has changed. In the response, the fDeleted will be set to true on the returned Contact for "tombstoned" (deleted) Contacts. fDeleted will be set to false on the returned Contact for added or updated Contacts.

There is are two cases where requests for deltas will fail.

1) Deleted Contacts remain in the Address Book for a fixed period of time (set to 90 days currently). Requests for deltas with a lastChange date longer than the interval will fail.

2) If an Address Book creation date is later than the lastChange date, the Address Book will assume the AB has been deleted and re-created. In that case, the call will fail.

If deltasOnly false, and the lastChange is set in the filter, the entire list will be returned only if there have been changes since lastChange.

[in] lastChange

Format: GMT. ISO 8601 format. Purpose: Used by partners that keep a local cache of Groups and Contacts and need to synchronize with the Address Book. See the section "Handling lastChange" in the Contacts API document for more information.

[return] FindResult

See the section on FindResult in ABFindAll for more information.

Table:SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABFindAll"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABFindAll
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <abView>
                    Full or
                    Hotmail or
                    MessengerServer or
```

```
                    MessengerClient
                </abView>
                <deltasOnly>boolean</deltasOnly>
                <lastChange>dateTime</lastChange>
            </ABFindAll>
        </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABFindAllResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
                <ABFindAllResult>
                     <ab>
                          <abId>guid</abId>
                          <abInfo>
                               <name>string</name>
                               <ownerPuid>long</ownerPuid>
                               <ownerEmail>string</ownerEmail>
                               <fDefault>boolean</fDefault>

     <isMessengerMigrated>boolean</isMessengerMigrated>
                          </abInfo>
                          <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                          <createDate>dateTime</createDate>
                          <lastChange>dateTime</lastChange>
                     </ab>
                     <contacts>
                          <Contact>
                               <contactId>guid</contactId>
                               <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                               <fDeleted>boolean</fDeleted>
                               <lastChange>dateTime</lastChange>
                          </Contact>
```

```
                    <Contact>
                        <contactId>guid</contactId>
                        <contactInfo xsi:nil="true" />

    <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                        <fDeleted>boolean</fDeleted>
                        <lastChange>dateTime</lastChange>
                    </Contact>
                </contacts>
                <groups>
                    <Group>
                        <groupId>guid</groupId>
                        <groupInfo xsi:nil="true" />
                        <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                        <fDeleted>boolean</fDeleted>
                        <lastChange>dateTime</lastChange>
                    </Group>
                    <Group>
                        <groupId>guid</groupId>
                        <groupInfo xsi:nil="true" />
                        <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                        <fDeleted>boolean</fDeleted>
                        <lastChange>dateTime</lastChange>
                    </Group>
                </groups>
            </ABFindAllResult>
        </ABFindAllResponse>
    </soap:Body>
</soap:Envelope>
```

C(ix)b. FindResult

FindResult and Description
.groups: All groups in the Address Book. The Groups are ordered by Group ID.

.contacts: All contacts in the Address Book. Both SMTP and Rich Contacts are returned. The Group membership of each Contact is returned.

.ab: Address Book properties

C(ix)c. ABView

ABView specifies the contact and group properties to return.

public enum ABView

```
{
        Full = 0,
        Hotmail = 1,
        MessengerServer = 2,
        MessengerClient = 3
}
```

Hotmail:

Contact Properties: ContactBirthDate, ContactEmail, ContactFirstName, ContactLastName, ContactPrimaryEmailType, ContactQuickName, ContactQuickOrder, IsFavorite, IsMessengerUser, IsSmtp, Passport, ContactPhone, ContactLocation

Group Properties: GroupName and QuickOrder

MessengerServer:

Contact Properties: Passport, IsMobileIMEnabled, IsMessengerUser, SpotWatchState, ContactPhone and Annotation

Group Properties: GroupName andQuickOrder

MessengerClient:

Contact Properties: ContactQuickName, Passport, ContactFirstName, ContactLastName, IsMessengerUser, ContactPrimaryEmailType, ContactEmail, IsMobileIMEnabled and SpotWatchState

Group Properties: GroupName and QuickOrder

C(x). ABFindByContacts

This method returns all of the Contacts with the given contact IDs as well as all of the Groups that contain those Contacts.

C(x)a. ABFindByContacts

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public FindResult ABFindByContacts(Guid abId, ABView abView, Guid[] contactIds)
```

Parameters

[hdr] Auth Header and Application Header

See the SOAP Header section in the Contacts API spec for more information.

[in] abId

Find Contacts in this Address Book.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] abView

See the section on ABView in ABFindAll for more information.

[in] contactIds

The list of specific Contacts to find.

[return] FindResult

See the section on FindResult in ABFindAll for more information.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABFindByContacts"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABFindByContacts
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <abView>Full or Hotmail or MessengerServer or
MessengerClient</abView>
               <contactIds>
                    <guid>guid</guid>
                    <guid>guid</guid>
```

```
            </contactIds>
        </ABFindByContacts>
    </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ABFindByContactsResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
            <ABFindByContactsResult>
                <ab>
                    <abId>guid</abId>
                    <abInfo>
                        <name>string</name>
                        <ownerPuid>long</ownerPuid>
                        <ownerEmail>string</ownerEmail>
                        <fDefault>boolean</fDefault>

    <isMessengerMigrated>boolean</isMessengerMigrated>
                    </abInfo>
                    <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                    <createDate>dateTime</createDate>
                    <lastChange>dateTime</lastChange>
                </ab>
                <contacts>
                    <Contact>
                        <contactId>guid</contactId>
                        <contactInfo xsi:nil="true" />

   <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                        <fDeleted>boolean</fDeleted>
                        <lastChange>dateTime</lastChange>
                    </Contact>
                    <Contact>
                        <contactId>guid</contactId>
                        <contactInfo xsi:nil="true" />
```

```
      <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Contact>
                    </contacts>
                    <groups>
                         <Group>
                              <groupId>guid</groupId>
                              <groupInfo xsi:nil="true" />
                              <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Group>
                         <Group>
                              <groupId>guid</groupId>
                              <groupInfo xsi:nil="true" />
                              <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Group>
                    </groups>
               </ABFindByContactsResult>
          </ABFindByContactsResponse>
     </soap:Body>
</soap:Envelope>
```

C(xi). ABFindByGroups

Returns all of the contacts in the given groups and the groups themselves.

C(xi)a. ABFindByGroups

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public FindResult ABFindByGroups( Guid abId, ABView abView, Guid[] groupIds)
```

Parameters

[hdr] Auth Header and Application Header

See the SOAP Header section in the Contacts API spec for more information.

[in] abId

Find Contacts in this Address Book.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] abView

See the section on ABView in ABFindAll for more information.

[in] groupIds

The list of specific Groups to find.

[return] FindResult

See the section on FindResult in ABFindAll for more information.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABFindByGroups"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABFindByGroups
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
```

```
                <abView>Full or Hotmail or MessengerServer or
MessengerClient</abView>
                <groupIds>
                     <guid>guid</guid>
                     <guid>guid</guid>
                </groupIds>
           </ABFindByGroups>
     </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABFindByGroupsResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
                <ABFindByGroupsResult>
                     <ab>
                          <abId>guid</abId>
                          <abInfo>
                               <name>string</name>
                               <ownerPuid>long</ownerPuid>
                               <ownerEmail>string</ownerEmail>
                               <fDefault>boolean</fDefault>

     <isMessengerMigrated>boolean</isMessengerMigrated>
                          </abInfo>
                          <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                          <createDate>dateTime</createDate>
                          <lastChange>dateTime</lastChange>
                     </ab>
                     <contacts>
                          <Contact>
                               <contactId>guid</contactId>
                               <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                               <fDeleted>boolean</fDeleted>
```

```
                              <lastChange>dateTime</lastChange>
                         </Contact>
                         <Contact>
                              <contactId>guid</contactId>
                              <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Contact>
                    </contacts>
                    <groups>
                         <Group>
                              <groupId>guid</groupId>
                              <groupInfo xsi:nil="true" />
                              <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Group>
                         <Group>
                              <groupId>guid</groupId>
                              <groupInfo xsi:nil="true" />
                              <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Group>
                    </groups>
               </ABFindByGroupsResult>
          </ABFindByGroupsResponse>
     </soap:Body>
</soap:Envelope>
```

C(xii). ABFindMessengerUsers

Returns all Contacts that have the isMessengerUser flag set - regardless of the Group membership. Also returns the "Me" contact and all Groups that contain any of the returned Contacts.

C(xii)a. ABFindMessengerUsers

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public FindResult ABFindMessengerUsers(Guid abId, ABView abView)
```

Parameters

[hdr] Auth Header and Application Header

See the SOAP Header section in the Contacts API spec for more information.

[in] abId

Find Contacts in this Address Book.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] abView

See the section on ABView in ABFindAll for more information.

[return] FindResult

See the section on FindResult in ABFindAll for more information.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABFindMessengerUser
s"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
```

```
          <ABFindMessengerUsers
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <abView>Full or Hotmail or MessengerServer or
MessengerClient</abView>
          </ABFindMessengerUsers>
     </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABFindMessengerUsersResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ABFindMessengerUsersResult>
                    <ab>
                         <abId>guid</abId>
                         <abInfo>
                              <name>string</name>
                              <ownerPuid>long</ownerPuid>
                              <ownerEmail>string</ownerEmail>
                              <fDefault>boolean</fDefault>

     <isMessengerMigrated>boolean</isMessengerMigrated>
                         </abInfo>
                         <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                         <createDate>dateTime</createDate>
                         <lastChange>dateTime</lastChange>
                    </ab>
                    <contacts>
                         <Contact>
                              <contactId>guid</contactId>
                              <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
```

```
                    </Contact>
                    <Contact>
                         <contactId>guid</contactId>
                         <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Contact>
               </contacts>
               <groups>
                    <Group>
                         <groupId>guid</groupId>
                         <groupInfo xsi:nil="true" />
                         <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Group>
                    <Group>
                         <groupId>guid</groupId>
                         <groupInfo xsi:nil="true" />
                         <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Group>
               </groups>
          </ABFindMessengerUsersResult>
     </ABFindMessengerUsersResponse>
</soap:Body>
</soap:Envelope>
```

C(xiii). ABFindMeContact

Returns only the "Me" contact.

Annotations are returned as well if relevant to the abView.

C(xiii)a. ABFindMeContact

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public FindResult ABFindMeContact(    Guid abId, ABView abView)
```

Parameters

[hdr] Auth Header and Application Header

See the SOAP Header section in the Contacts API spec for more information.

[in] abId

Find Contacts in this Address Book.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] abView

See the section on ABView in ABFindAll for more information.

[return] FindResult

See the section on FindResult in ABFindAll for more information.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABFindMeContact"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABFindMeContact
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
```

```
                <abView>Full or Hotmail or MessengerServer or
MessengerClient</abView>
          </ABFindMeContact>
     </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABFindMeContactResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ABFindMeContactResult>
                    <ab>
                         <abId>guid</abId>
                         <abInfo>
                              <name>string</name>
                              <ownerPuid>long</ownerPuid>
                              <ownerEmail>string</ownerEmail>
                              <fDefault>boolean</fDefault>

     <isMessengerMigrated>boolean</isMessengerMigrated>
                         </abInfo>
                         <propertiesChanged>Name or
IsMessengerMigrated</propertiesChanged>
                         <createDate>dateTime</createDate>
                         <lastChange>dateTime</lastChange>
                    </ab>
                    <contacts>
                         <Contact>
                              <contactId>guid</contactId>
                              <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Contact>
                         <Contact>
                              <contactId>guid</contactId>
```

```
                              <contactInfo xsi:nil="true" />

     <propertiesChanged>ContactPrimaryEmailType or ContactFirstName
or ContactLastName or ContactQuickName or ContactBirthDate or
ContactQuickOrder or ContactEmail or ContactPhone or ContactLocation
or ContactWebSite or Annotation or Passport or Comment or
IsMobileVisible or IsMobileIMEnabled or IsMessengerUser or
IsFavorite or IsSmtp or SpotWatchState</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Contact>
                    </contacts>
                    <groups>
                         <Group>
                              <groupId>guid</groupId>
                              <groupInfo xsi:nil="true" />
                              <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Group>
                         <Group>
                              <groupId>guid</groupId>
                              <groupInfo xsi:nil="true" />
                              <propertiesChanged>GroupName or
Annotation or QuickOrder</propertiesChanged>
                              <fDeleted>boolean</fDeleted>
                              <lastChange>dateTime</lastChange>
                         </Group>
                    </groups>
               </ABFindMeContactResult>
          </ABFindMeContactResponse>
     </soap:Body>
</soap:Envelope>
```

C(xiv). ABGroupAdd

Add a Group or set of Groups to the Address Book.

C(xiv)a. ABGroupAdd

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public Guid[] ABGroupAdd(Guid abId,  GroupInfo[] groupInfo)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the appendix.

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Identifies the Address Book where the Group will be added.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] groupInfo

One or more Groups to be added. See Bulk Limits for more info. A null array entry will give a NullArgument exception.

.groupType

MSNAB (default)
MSNBUDDY
MSNPUBLIC

.name

Group name - must be unique within Group type.

..clientErrorData

Include a client determined value for this field when adding multiple Groups. This value will be returned as the identity of the Group should that Group fail the Add for any reason.

[return] Guid

Id's for the Groups created.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABGroupAdd"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
```

```
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABGroupAdd
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <groupInfo>
                    <GroupInfo>
                         <groupType>guid</groupType>
                         <name>string</name>
                         <quickOrder>short</quickOrder>
                         <annotations>
                              <Annotation xsi:nil="true" />
                              <Annotation xsi:nil="true" />
                         </annotations>
                         <clientErrorData>string</clientErrorData>
                    </GroupInfo>
                    <GroupInfo>
                         <groupType>guid</groupType>
                         <name>string</name>
                         <quickOrder>short</quickOrder>
                         <annotations>
                              <Annotation xsi:nil="true" />
                              <Annotation xsi:nil="true" />
                         </annotations>
                         <clientErrorData>string</clientErrorData>
                    </GroupInfo>
               </groupInfo>
          </ABGroupAdd>
     </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
```

```
            <ABGroupAddResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
                  <ABGroupAddResult>
                        <guid>guid</guid>
                        <guid>guid</guid>
                  </ABGroupAddResult>
            </ABGroupAddResponse>
      </soap:Body>
</soap:Envelope>
```

C(xiv)b. Additional Notes

Note: Deleted Groups or Contacts are actually "tombstoned" in the ABCH. The deleted entry remains in the database marked as fDeleted=true. If another Group or Contact is created with the same Group Name or Quickname, the existing entry is "resurrected" with fields cleared and replaced by the new entry.

There is no umbrella transaction over bulk Group Add operations. If one of the Groups fails to add, all Adds prior to that failure will be committed. Any Group adds after the failure are discarded and must be retried by the client.

C(xiv)c. Return Id on Conflict

When adding a Group whose name conflicts with another Group, the Id of the conflicting Group will be returned.

See the section on Return Id on Conflict for more information.

C(xv). ABGroupDelete

Delete a Group or set of Groups from the Address Book.

C(xv)a. ABGroupDelete

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public void ABGroupDelete(Guid abId,  GroupFilter groupFilter)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the appendix.

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Identifies the Address Book

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] groupFilter

Identifies one or more Groups to delete. You can pass groupsIds or groupTypes but not both in one call. See Bulk Limits for more info.

.groupIds List of Group Ids to delete.

You cannot delete the MSNAB Group or MSNBUDDY Group. Returns GroupCannotBeDeleted.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABGroupDelete"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABGroupDelete
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <groupFilter>
                    <groupIds>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupIds>
                    <groupTypes>
                         <guid>guid</guid>
                         <guid>guid</guid>
```

```
                    </groupTypes>
                </groupFilter>
            </ABGroupDelete>
        </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABGroupDeleteResponse
xmlns="http://contacts.msn.com/webservices/AddressBook" />
     </soap:Body>
</soap:Envelope>
```

C(xv)b. Additional Notes

Note: Deleted Groups or Contacts are actually "tombstoned" in the ABCH. The deleted entry remains in the database marked as fDeleted=true. If another Group or Contact is created with the same Group Name or Quickname, the existing entry is "resurrected" with fields cleared and replaced by the new entry.

However if you try to rename another group to this name, you will get the "Group Already Exists" error. The ABCH cleans up tombstones every n days (currently 90 days) - so you won't be able to rename another group to a tombstoned group until that period has elapsed.

Bulk Group deletes are transacted. If one Group fails to delete, all the deletions are rolled back.

C(xvi). ABGroupUpdate

Update properties for a Group or set of Groups in the Address Book.

C(xvi)a. ABGroupUpdate

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public void ABGroupUpdate(Guid abId, Group[] groups)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the document.

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Target Address Book.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] groups

One or more Groups to be updated. See Bulk Limits for more info. A null array entry will give a NullArgument exception.

.groupId Required. Id of the Group to be changed.

.groupInfo.groupType
Cannot be updated.

.groupInfo.name
New Group name. Must be UNIQUE within Group type.

.groupInfo.clientErrorData
Include a client determined value for this field when updating multiple Groups. This value will be returned as the identity of the Group should that Group fail the update for any reason.

.propertiesChanged
Set by the caller to indicate which fields should be updated. Required. See GroupPropertyType.
GroupName

.lastChange
.fDeleted
Cannot be updated.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABGroupUpdate"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
```

```
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABGroupUpdate
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <groups>
                    <Group>
                         <groupId>guid</groupId>
                         <groupInfo>
                              <groupType>guid</groupType>
                              <name>string</name>
                              <quickOrder>short</quickOrder>
                              <annotations xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </groupInfo>
                         <propertiesChanged>GroupName or Annotation
or QuickOrder</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Group>
                    <Group>
                         <groupId>guid</groupId>
                         <groupInfo>
                              <groupType>guid</groupType>
                              <name>string</name>
                              <quickOrder>short</quickOrder>
                              <annotations xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </groupInfo>
                         <propertiesChanged>GroupName or Annotation
or QuickOrder</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Group>
               </groups>
          </ABGroupUpdate>
     </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABGroupUpdateResponse
xmlns="http://contacts.msn.com/webservices/AddressBook" />
     </soap:Body>
</soap:Envelope>
```

C(xvi)b. Additional Notes

There is no umbrella transaction over bulk Group Update operations. If one of the Groups fails to update, all updates prior to that failure will be committed. Any Group updates after the failure are discarded and must be retried by the client.

C(xvi)c. Return Id on Conflict

When adding a Group whose name conflicts with another Group, the Id of the conflicting Group will be returned.

See the section on Return Id on Conflict for more information.

C(xvii). ABGroupFind

Find all Groups in the Address Book. This returns just the Group information and not the Contacts in the Group. Tombstoned Groups are also returned with the call.

C(xvii)a. ABGroupFind

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public Group[] ABGroupFind(Guid abId, GroupFilter groupFilter, bool getAnnotations,
Annotation[] annotations)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the document.

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Identifies the Address Book that contains the Groups.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] groupFilter

Criteria for the search. See the GroupFilter table for more information.

If the groupFilter is null, then all the Groups should be returned. This includes all tombstoned Groups.

If the groupFilter not null, either groupTypes or groupIds MUST be supplied.

If a Guid provided in the groupIds does not exist, or if the groupType is invalid, no error will result. The result set may in fact be empty if only non-existing or invalid types are specified.

If both groupIds and groupTypes are given in the same request, the filter is an OR of these values, and the union of the result set will be returned.

.groupIds List of Group Ids to find.

.groupTypes List of Group types to find. Must be the Guids for these Group Types:
MSNAB
MSNBUDDY
MSNPUBLIC

[in] getAnnotations

If true, annotations are returned with the call.

[in] annotations

Specify the annotations that should appear in the response.

[return] Groups

Result set of Groups that match the query. There is no limit to the number of Groups that may be returned in a request.

Groups that have been deleted from the AB will also be returned with the fDeleted flag set to true.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABGroupFind"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABGroupFind
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <groupFilter>
                    <groupIds>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupIds>
                    <groupTypes>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupTypes>
               </groupFilter>
               <getAnnotations>boolean</getAnnotations>
               <annotations>
                    <Annotation>
                         <Name>string</Name>
                         <Value>string</Value>
                    </Annotation>
                    <Annotation>
                         <Name>string</Name>
                         <Value>string</Value>
                    </Annotation>
               </annotations>
          </ABGroupFind>
     </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABGroupFindResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ABGroupFindResult>
                    <Group>
                         <groupId>guid</groupId>
                         <groupInfo>
                              <groupType>guid</groupType>
                              <name>string</name>
                              <quickOrder>short</quickOrder>
                              <annotations xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </groupInfo>
                         <propertiesChanged>GroupName or Annotation
or QuickOrder</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Group>
                    <Group>
                         <groupId>guid</groupId>
                         <groupInfo>
                              <groupType>guid</groupType>
                              <name>string</name>
                              <quickOrder>short</quickOrder>
                              <annotations xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </groupInfo>
                         <propertiesChanged>GroupName or Annotation
or QuickOrder</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Group>
               </ABGroupFindResult>
          </ABGroupFindResponse>
     </soap:Body>
</soap:Envelope>
```

C(xviii). ABGroupContactAdd

This method is used to:

- Add new Contacts to Groups.

- Merge new Contacts into Groups.
- Add an existing Contact to a Group (change in membership).

C(xviii)a. ABGroupContactAdd

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
public Guid[] ABGroupContactAdd(Guid abId, GroupFilter groupFilter, Contact[]
contacts,GroupContactAddOptions groupContactAddOptions)
```

Parameters

Information specific to this method is listed here. The rest of the information on the fields can be found in the Group/Contact sections at the beginning of the appendix.

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Adds the Contacts to Groups in this Address Book.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] groupFilter

Adds the Contacts to the Groups specified in this filter. The Groups can be identified by their groupId, groupType, or both. At least one Group MUST be specified.

.groupIds The specified Contacts should go into these individual Groups.

.groupTypes The specified Contacts should go into these Group types.
MSNAB
MSNBUDDY
Note: MSNPUBLIC is not a valid groupType for ABGroupContactAdd. There is no mechanism to add a Contact to all MSNPUBLIC Groups.

[in] contacts

One or more Contacts to add. See Bulk Limits for more info. A null array entry will give a NullArgument exception.

.contactId

Do not supply a Contact ID when creating a Contact.

Specify the Contact Id when adding an existing Contact to a new Group or set of Groups. This is a Group membership change only.

.contactInfo.quickName

Do not supply a quickName or any other field when adding an existing Contact to a Group.

When adding a new Contact, the Contact quickName is a required and unique field in the ABCH. If you do not supply a Quickname, you MUST set the fGenerateMissingQuickName flag on this call.

.contactInfo.isSmtp

Set to true when a Contact is representing an SMTP address only. Equivalent to Outlook SMTP Contact. SMTP Contacts cannot be added to the MSNAB Group.

.contactInfo.firstName/.lastName
.contactInfo.birthdate
.contactInfo.quickOrder
.contactInfo.primaryEmailType

See Contact table for more information.

.contactInfo.emails
.contactInfo.phones
.contactInfo.locations
.contactInfo.websites

Note that within each array, only one occurrence of each subtype is allowed. For example, within the emails array, an email of type ContactEmailPersonal can only appear once in the array.

See Contact table for more information.

.contactInfo.groupIds
contactInfo.groupIdsDeleted

Not used during Add.

.contactInfo.clientErrorData

Include a client determined value for this field when Adding multiple Contacts. This value will be returned as the identity of the Contact should that Contact fail the Add for any reason.

.propertiesChanged

.fDeleted

Ignored on Add operations.

.lastChange

Ignored on Add operations.

[in] groupContactAddOptions

.fGenerateMissingQuickName

true/false. This flag must be set if the Quickname should be generated by the ABCH.

[return] Guid
The list of identities for the specified Contacts is returned.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABGroupContactAdd"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABGroupContactAdd
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <groupFilter>
                    <groupIds>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupIds>
                    <groupTypes>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupTypes>
               </groupFilter>
               <contacts>
                    <Contact>
                         <contactId>guid</contactId>
                         <contactInfo>
```

```
                              <contactType>Regular or
Me</contactType>
                              <quickName>string</quickName>
                              <firstName>string</firstName>
                              <lastName>string</lastName>
                              <passportName>string</passportName>
                              <puid>long</puid>
                              <comment>string</comment>

     <isMobileVisible>boolean</isMobileVisible>

     <isMobileIMEnabled>boolean</isMobileIMEnabled>

     <isMessengerUser>boolean</isMessengerUser>
                              <isFavorite>boolean</isFavorite>
                              <isSmtp>boolean</isSmtp>
                              <spotWatchState>NoDevice or
NoMessaging or MessagingEnabled</spotWatchState>
                              <birthdate>dateTime</birthdate>
                              <quickOrder>short</quickOrder>
                              <primaryEmailType>ContactEmailPersonal
or ContactEmailBusiness or ContactEmailOther or
ContactEmailMessenger</primaryEmailType>
                              <emails xsi:nil="true" />
                              <phones xsi:nil="true" />
                              <locations xsi:nil="true" />
                              <webSites xsi:nil="true" />
                              <annotations xsi:nil="true" />
                              <groupIds xsi:nil="true" />
                              <groupIdsDeleted xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </contactInfo>
                         <propertiesChanged>ContactPrimaryEmailType
or ContactFirstName or ContactLastName or ContactQuickName or
ContactBirthDate or ContactQuickOrder or ContactEmail or
ContactPhone or ContactLocation or ContactWebSite or Annotation or
Passport or Comment or IsMobileVisible or IsMobileIMEnabled or
IsMessengerUser or IsFavorite or IsSmtp or
SpotWatchState</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Contact>
                    <Contact>
                         <contactId>guid</contactId>
                         <contactInfo>
                              <contactType>Regular or
Me</contactType>
                              <quickName>string</quickName>
```

```
                              <firstName>string</firstName>
                              <lastName>string</lastName>
                              <passportName>string</passportName>
                              <puid>long</puid>
                              <comment>string</comment>

     <isMobileVisible>boolean</isMobileVisible>

     <isMobileIMEnabled>boolean</isMobileIMEnabled>

     <isMessengerUser>boolean</isMessengerUser>
                              <isFavorite>boolean</isFavorite>
                              <isSmtp>boolean</isSmtp>
                              <spotWatchState>NoDevice or
NoMessaging or MessagingEnabled</spotWatchState>
                              <birthdate>dateTime</birthdate>
                              <quickOrder>short</quickOrder>
                              <primaryEmailType>ContactEmailPersonal
or ContactEmailBusiness or ContactEmailOther or
ContactEmailMessenger</primaryEmailType>
                              <emails xsi:nil="true" />
                              <phones xsi:nil="true" />
                              <locations xsi:nil="true" />
                              <webSites xsi:nil="true" />
                              <annotations xsi:nil="true" />
                              <groupIds xsi:nil="true" />
                              <groupIdsDeleted xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </contactInfo>
                         <propertiesChanged>ContactPrimaryEmailType
or ContactFirstName or ContactLastName or ContactQuickName or
ContactBirthDate or ContactQuickOrder or ContactEmail or
ContactPhone or ContactLocation or ContactWebSite or Annotation or
Passport or Comment or IsMobileVisible or IsMobileIMEnabled or
IsMessengerUser or IsFavorite or IsSmtp or
SpotWatchState</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Contact>
               </contacts>
               <groupContactAddOptions>
                    <fGenerateMissingQuickName>
                         boolean
                    </fGenerateMissingQuickName>
               </groupContactAddOptions>
          </ABGroupContactAdd>
     </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABGroupContactAddResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ABGroupContactAddResult>
                    <guid>guid</guid>
                    <guid>guid</guid>
               </ABGroupContactAddResult>
          </ABGroupContactAddResponse>
     </soap:Body>
</soap:Envelope>
```

C(xviii)b. Additional Notes

Note: Deleted Groups or Contacts are actually "tombstoned" in the ABCH. The deleted entry remains in the database marked as fDeleted=true. If another Group or Contact is created with the same Group Name or Quickname, the existing entry is "resurrected" with fields cleared and replaced by the new entry.

There is no umbrella transaction over bulk Contact Add operations. If one of the Contacts fails to add, all Adds prior to that failure MAY be committed. The client must perform an fDeltasOnly synchronization to determine which Contacts had committed. Any Contact Adds after the failure are discarded and must be retried by the client.

C(xviii)c. Adding to MSNAB adds to all Contacts (Everyone)

Adding a new Contact to the MSNAB Group is the same as adding to the list of all Contacts. The MSNAB Group is not maintained.

C(xviii)d. Add Contact Parental Controls

See the section on Parental Controls for more information.

Through the Passport auth'ed Front End, Managed accounts will not be allowed to Add a Messenger User.

C(xviii)e. Return Id on Conflict

When adding a Contact whose quickname or passportName conflicts with another Contact, the Id of the conflicting Contact will be returned.

See the section on Return Id on Conflict for more information.

C(xviii)f. contactType, isSmtp, isFavorite defaults

If contactType is NOT specified, the contactType will default to Regular.

If isSmtp is not specified, the behavior should be as follows:

- If the Contact is being added to a Non-MSNAB Group, and only a personal email address is specified, then an SMTP Contact will be created. Also note, isFavorite will be set to false in this case. If other fields are specified, a Rich Contact will be created.
- If the Contact is being added to the MSNAB Group, then a Rich Contact will be created, and isFavorite will be set to true.

C(xviii)g. fGenerateMissingQuickname

Default basename selection: The Passport field will be the first choice when choosing a basename. If no Passport field exists, the remaining algorithm is used (try the primary email address, work, other, first/last, etc).

For Rich Contacts, the basename is placed into the Quickname, and if the basename is not unique, only then are numbers appended to make the Quickname unique.

For SMTP Contacts, CHANGE in logic, the basename is always obscured. The default quickname will be the basename plus 3 digits. If this name is not unique (very low probability), the backend will make it unique by attempting to append 2 additional digits.

Optionally, when the backend attempts to find a unique name, it will retry up to 1000 times. As an optimization, if the backend cannot find a unique quickname after both the 2 digit and 4 digit attempts (10 each), a GUID should be placed in the Quickname rather than continuing to attempt to make unique.

C(xviii)h. isSmtp validation

If isSmtp = true,

- Only ContactEmailPersonal can be set
- Primary email cannot point to anything accept ContactEmailPersonal
- First and Last name may be set
- No other fields may be set.
- fGenerateMissingQuickname MUST be true

C(xix). ABGroupContactDelete

Delete a Contact from a Group or set of Groups.

C(xix)a. ABGroupContactDelete

```
[SoapHeader("m_abAuthHeader", Required=false)]
[SoapHeader("m_abAppHeader", Required=true)]
```

public void ABGroupContactDelete(Guid abId, GroupFilter groupFilter, Contact[] contacts// Added 2-28-02)

Parameters

[hdr] Auth Header and Application Header

See the SOAP Header section for more information.

[in] abId

Identifies the Address Book that contains the Group.

For clients using the Passport authed FE, the abId may be zero. This will cause the Address Book to get the PUID from the authentication header in the SOAP request.

[in] groupFilter

Identifies one or more Groups in which to delete the Contact. See Bulk Limits for more info.

.groupIds The specified Contacts should be deleted from these individual Groups.

.groupTypes The specified Contacts should be deleted from these Group types.
MSNAB
MSNBUDDY

groupIds and groupTypes can be given simultaneously in ABGroupContactDelete. (DCR 9665)

[in] contacts

Identifies the Contact to delete from the Group. Contact can be identified by Id only.

.contactId
Contact ID for the Contact to be removed from the specified Groups.

.contactInfo
MUST be null.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABGroupContactDelet
e"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABGroupContactDelete
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <groupFilter>
                    <groupIds>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupIds>
                    <groupTypes>
                         <guid>guid</guid>
                         <guid>guid</guid>
                    </groupTypes>
               </groupFilter>
               <contacts>
                    <Contact>
                         <contactId>guid</contactId>
                         <contactInfo>
                              <contactType>Regular or
Me</contactType>
                              <quickName>string</quickName>
                              <firstName>string</firstName>
                              <lastName>string</lastName>
                              <passportName>string</passportName>
                              <puid>long</puid>
                              <comment>string</comment>

     <isMobileVisible>boolean</isMobileVisible>

     <isMobileIMEnabled>boolean</isMobileIMEnabled>

     <isMessengerUser>boolean</isMessengerUser>
                              <isFavorite>boolean</isFavorite>
```

```
                              <isSmtp>boolean</isSmtp>
                              <spotWatchState>NoDevice or
NoMessaging or MessagingEnabled</spotWatchState>
                              <birthdate>dateTime</birthdate>
                              <quickOrder>short</quickOrder>
                              <primaryEmailType>ContactEmailPersonal
or ContactEmailBusiness or ContactEmailOther or
ContactEmailMessenger</primaryEmailType>
                              <emails xsi:nil="true" />
                              <phones xsi:nil="true" />
                              <locations xsi:nil="true" />
                              <webSites xsi:nil="true" />
                              <annotations xsi:nil="true" />
                              <groupIds xsi:nil="true" />
                              <groupIdsDeleted xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                         </contactInfo>
                         <propertiesChanged>ContactPrimaryEmailType
or ContactFirstName or ContactLastName or ContactQuickName or
ContactBirthDate or ContactQuickOrder or ContactEmail or
ContactPhone or ContactLocation or ContactWebSite or Annotation or
Passport or Comment or IsMobileVisible or IsMobileIMEnabled or
IsMessengerUser or IsFavorite or IsSmtp or
SpotWatchState</propertiesChanged>
                         <fDeleted>boolean</fDeleted>
                         <lastChange>dateTime</lastChange>
                    </Contact>
                    <Contact>
                         <contactId>guid</contactId>
                         <contactInfo>
                              <contactType>Regular or
Me</contactType>
                              <quickName>string</quickName>
                              <firstName>string</firstName>
                              <lastName>string</lastName>
                              <passportName>string</passportName>
                              <puid>long</puid>
                              <comment>string</comment>

     <isMobileVisible>boolean</isMobileVisible>

     <isMobileIMEnabled>boolean</isMobileIMEnabled>

     <isMessengerUser>boolean</isMessengerUser>
                              <isFavorite>boolean</isFavorite>
                              <isSmtp>boolean</isSmtp>
                              <spotWatchState>NoDevice or
NoMessaging or MessagingEnabled</spotWatchState>
```

```
                                <birthdate>dateTime</birthdate>
                                <quickOrder>short</quickOrder>
                                <primaryEmailType>ContactEmailPersonal
or ContactEmailBusiness or ContactEmailOther or
ContactEmailMessenger</primaryEmailType>
                                <emails xsi:nil="true" />
                                <phones xsi:nil="true" />
                                <locations xsi:nil="true" />
                                <webSites xsi:nil="true" />
                                <annotations xsi:nil="true" />
                                <groupIds xsi:nil="true" />
                                <groupIdsDeleted xsi:nil="true" />

     <clientErrorData>string</clientErrorData>
                           </contactInfo>
                           <propertiesChanged>ContactPrimaryEmailType
or ContactFirstName or ContactLastName or ContactQuickName or
ContactBirthDate or ContactQuickOrder or ContactEmail or
ContactPhone or ContactLocation or ContactWebSite or Annotation or
Passport or Comment or IsMobileVisible or IsMobileIMEnabled or
IsMessengerUser or IsFavorite or IsSmtp or
SpotWatchState</propertiesChanged>
                           <fDeleted>boolean</fDeleted>
                           <lastChange>dateTime</lastChange>
                      </Contact>
                 </contacts>
            </ABGroupContactDelete>
     </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABGroupContactDeleteResponse
xmlns="http://contacts.msn.com/webservices/AddressBook" />
     </soap:Body>
</soap:Envelope>
```

C(xix)b. Additional Notes

If the Contact is not contained in any other Groups after the requested delete, then the Contact will be deleted from the Address Book.

Note: Deleted Groups or Contacts are actually "tombstoned" in the ABCH. The deleted entry remains in the database marked as fDeleted=true. If another Group or Contact is created with the same Group Name or Quickname, the existing entry is "resurrected" with fields cleared and replaced by the new entry.

Note: The Group membership is also "tombstoned" in the database. This is the mechanism used to determine deltas for members of Groups. The membership table in the ABCH database carries a separate last modified date and deleted flag for each Contact/Group membership row.

Bulk Contact deletes are transacted. If one Contact fails to delete, all the deletions are rolled back.

C(xix)c. MSNAB / MSNPUBLIC Handing

Deletion from MSNAB deletes the Contact. Even if that Contact is referenced in multiple Groups. Deletion from MSNAB is equivalent to ABContactDelete.

Deletion of a Rich Contact from an MSNPUBLIC Group never deletes the Contact. The Contact is simply removed from the Group.

C(xix)d. Tombstones must have Null Fields

When deleting a Contact, the passportName and isMessengerUser field will be cleared. This insures the tombstone will never have a conflicting passportName field.

C(xx). ABAllowListSet

Set the Parental Control allow list in the Address Book. The Allow List is a string of smtp addresses that represent the persons the managed account may send and receive email.

C(xx)a. ABAllowListSet

```
[SoapHeader("m_abAuthHeader", Required=false)]
    [SoapHeader("m_abAppHeader", Required=true)]
    public void ABAllowListSet(Guid abId, string allowList)
```

Parameters

[hdr] Auth Header and Application Header

The <ApplicationId> MUST be from client application that generates compatible IDs.

The cookies provided in the HTTP header are the cookies of the parent.

The <ManagedGroupRequest> header is REQUIRED to make this call. Must be true.

See the SOAP Header section for more information.

[in] abId

The Address Book ID <abId> in this request is the PUID of the child.

[in] allowList

Max 7800 characters. String of smtp addresses. The exact format of the string is determined by client application and is not parsed or validated by the Address Book.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABAllowListSet"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABAllowListSet
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <abId>guid</abId>
               <allowList>string</allowList>
          </ABAllowListSet>
     </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <ABAllowListSetResponse
xmlns="http://contacts.msn.com/webservices/AddressBook" />
```

```
    </soap:Body>
</soap:Envelope>
```

C(xx)b. Additional Notes

ABCH permits the allow list to be up to 7800 characters. These are all expected to be ascii characters since this represents a list of email addresses. With an average email address size of 40 chars, one can store upto 190 email addresses in this list. The ABCH does not parse this information in any way.

The Allow List can be set only on a managed account. Only the parent account can set the allow list for a child managed account. The child account can only read the allow list.

Guidelines for the parent to set the child's Address Book:

- The <ApplicationId> MUST be from client application (thru the ApplicationId SOAP header).
- The cookies provided in the HTTP header are the cookies of the parent.
- The <ManagedGroupRequest> flag MUST be set to true.
- The Address Book ID <abId> in this request is the PUID of the child.
- Calls to ABAllowListSet MUST be secure (HTTPS) because the PUID of the child is passed in the clear.

The lastModifiedDate timestamp on the AddressBook is updated everytime the allow list is changed. This will also trigger a HTTP POST notification to HM.

C(xxi). ABAllowListGet

Get the Parental Control allow list in the Address Book. The Allow List is a string of smtp addresses that represent the persons the managed account may send and receive email.

C(xxi)a. ABAllowListGet

```
[SoapHeader("m_abAuthHeader", Required=false)]
    [SoapHeader("m_abAppHeader", Required=true)]
    public string ABAllowListGet(   Guid abId)
```

Parameters

[hdr] Auth Header and Application Header

For the parent to Get the child Allow List, the <ApplicationId> MUST be from client application (thru the ApplicationId SOAP header).
For the child to Get the child Allow List, the <ApplicationId> can be any valid ABCH partner.

For the parent to Get the child Allow List, The cookies provided in the HTTP header are the cookies of the parent.
For the child to Get the child Allow List, the cookies provided in the <ABAuthHeader> are the cookies of the child.

For the parent to Get the child Allow List, the <ManagedGroupRequest> flag MUST be set to true.
For the child to Get the child Allow List, the <ManagedGroupRequest> is not required.

See the SOAP Header section for more information.

[in] abId

For the parent to Get the child Allow List, the Address Book ID <abId> is the PUID of the child.
For the child to Get the child Allow List, the Address Book ID <abId> is zero.

[return] allowList

String of smtp addresses. The exact format of the string is determined by client application and is not parsed by the Address Book.

Table: SOAP Protocol Request

```
POST /abservice/abservice.asmx HTTP/1.1
Host: contacts.msn.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://contacts.msn.com/webservices/AddressBook/ABAllowListGet"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Header>
          <ABApplicationHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ApplicationId>guid</ApplicationId>
               <IsMigration>boolean</IsMigration>
          </ABApplicationHeader>
          <ABAuthHeader
xmlns="http://contacts.msn.com/webservices/AddressBook">
               <ManagedGroupRequest>boolean</ManagedGroupRequest>
          </ABAuthHeader>
     </soap:Header>
     <soap:Body>
          <ABAllowListGet
xmlns="http://contacts.msn.com/webservices/AddressBook">
```

```
                <abId>guid</abId>
            </ABAllowListGet>
        </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
            <ABAllowListGetResponse
xmlns="http://contacts.msn.com/webservices/AddressBook">
                <ABAllowListGetResult>string</ABAllowListGetResult>
            </ABAllowListGetResponse>
        </soap:Body>
</soap:Envelope>
```

C(xxi)b. Additional Notes

ABCH permits the allow list to be up to 7800 characters. These are all expected to be ascii characters since this represents a list of email addresses. With an average email address size of 40 chars, one can store upto 190 email addresses in this list. The ABCH does not parse this information in any way.

The child managed account can only read the allow list, and cannot set it. The parent can both set and get the Allow List.

Guidelines for the parent to get the child's Address Book:

- The <ApplicationId> MUST be from client application (thru the ApplicationId SOAP header).
- The cookies provided in the HTTP header are the cookies of the parent.
- The <ManagedGroupRequest> flag MUST be set to true.
- The Address Book ID <abId> in this request is the PUID of the child.
- The parent call to ABAllowListGet MUST be secure (HTTPS) because the PUID of the child is passed in the clear.

Guidelines for the child to get the child's Address Book:

- The <ApplicationId> can be any ABCH partner.
- The cookies provided in the HTTP header are the cookies of the child.
- The <ManagedGroupRequest> is not required.
- The Address Book ID <abId> is zero. In this case, the ABCH gets the PUID from the child cookies.
- Calls to ABAllowListGet by the child do not require SSL.